UNIVERSIDADE DO VALE DO ITAJAÍ
CENTRO DE CIÊNCIAS TECNOLÓGICAS DA TERRA E DO MAR
CURSO DE CIÊNCIA DA COMPUTAÇÃO

VALIDAÇÃO DA BIBLIOTECA KINECTPROJECT ATRAVÉS DA IMPLEMENTAÇÃO DE UMA INTERFACE NATURAL AO PROJETO DA TRILHA DO MORRO DA CRUZ

por

Sergio Roberto Gorniski Júnior

Itajaí (SC), dezembro de 2013

UNIVERSIDADE DO VALE DO ITAJAÍ
CENTRO DE CIÊNCIAS TECNOLÓGICAS DA TERRA E DO MAR
CURSO DE CIÊNCIA DA COMPUTAÇÃO

VALIDAÇÃO DA BIBLIOTECA KINECTPROJECT ATRAVÉS DA IMPLEMENTAÇÃO DE UMA INTERFACE NATURAL AO PROJETO DA TRILHA DO MORRO DA CRUZ

Área de Computação Gráfica

por

Sergio Roberto Gorniski Júnior

Relatório apresentado à Banca Examinadora do Trabalho Técnico-científico de Conclusão do Curso de Ciência da Computação para análise e aprovação.
Orientadora: Dra. Ana Elisa Schmidt

Itajaí (SC), dezembro de 2013

AGRADECIMENTOS

Agradeço minha família pela oportunidade de ingressar na faculdade que me fez chegar até aqui.

Agradeço minha namorada Larissa pelo apoio dado, que sempre me motivou e nunca me deixou desanimar. Agradeço também pela paciência nos momentos difíceis por muitas vezes não poder estar presente.

Agradeço também a minha orientadora Ana Elisa por todo suporte e conselhos dados para o desenvolvimento deste trabalho.

RESUMO

GORNISKI, Sergio Roberto. **Validação da biblioteca KinectProject através da implementação de uma Interface Natural para o Projeto da Trilha do Morro da Cruz**. Itajaí, 2013. 92. Trabalho Técnico-científico de Conclusão de Curso (Graduação em Ciência da Computação) – Centro de Ciências Tecnológicas da Terra e do Mar, Universidade do Vale do Itajaí, Itajaí, 2013.

Partindo-se da interação via linhas de comando (CLI), passando-se por telas gráficas com menus e botões (GUI), e hoje já é possível utilizar gestos, poses e comandos de voz para interagir com o computador. Estas interfaces naturais (NUI) necessitam de um hardware, como o Kinect desenvolvido pela Microsoft, para a captura de movimentos e voz. Este trabalho apresenta uma proposta e implementação de uma interface natural (NUI), utilizando o dispositivo Kinect, ao um estudo de caso utilizando-se da biblioteca KinectProject, desenvolvida como trabalho de conclusão de curso pelo Bacharel em Ciência da Computação pela UNIVALI Gabriel Schade Cardoso. A biblioteca encapsula o SDK desenvolvido pela Microsoft para o uso do Kinect em diversas aplicações. Este projeto tem como objetivo verificar e validar as funcionalidades, desempenho e facilidade de uso da KinectProject para a implementação de uma NUI, utilizando-se do projeto da Trilha Virtual do Morro da Cruz como estudo de caso. Este Projeto da Trilha foi desenvolvido na ferramenta Unity, a qual se trata de um motor de jogos que não possui suporte para configuração de uma interface natural com o Kinect. Propõe-se também um conjunto de comandos naturais baseados em estudos de usabilidade e a avaliação, pela perspectiva dos usuários, da interface natural implementada.

Palavras-chave: NUI. Trilha do Morro da Cruz. Kinect. KinectProject. Validação.

ABSTRACT

Starting with command line interfaces, passing by graphics screens with menus and buttons (GUI), today it's possible to use gestures, poses and voice commands to interact with the computer. These natural interfaces (NUI) usually pair with non-conventional devices, like the Kinect developed by Microsoft, to capture movements and voice. This paper presents a proposal and an implementation of a natural interface, with the Kinect, to a study case using the KinectProject's API developed as a term paper by the Bachelor of Computer Science from UNIVALI Gabriel Schade Cardoso. This API encapsulates the SDK developed by Microsoft to use Kinect in different applications. This project has as objective to verify and validate the KinectProject's functionalities, performance and ease of use to implement a NUI, using the Morro da Cruz's Trail's Project as a study case. This trail's project was developed in Unity, which is a game engine that don't support configuration of a natural interface with Kinect. It also proposes a set of natural commands based on usability studies and the evaluation, by the user's perspective, of the natural interface implemented.

Keywords: *NUI. Morro da Cruz's Trail. KinectProject. Validation.*

LISTA DE FIGURAS

Figura 1. Passeando pelo Projeto da Trilha do Morro da Cruz.....13
Figura 2. Principais interfaces de usuário.....17
Figura 3. Imagem do prompt de comando do Windows 8 18
Figura 4. Interface gráfica do SO Windows 1.0. 19
Figura 5. Zoom em uma imagem. 20
Figura 6. Imagem do Kinect adaptada.....21
Figura 7. Estrutura do SDK 22
Figura 8. Estrutura do KinectServer. 24
Figura 9. Estrutura do InputRegisterApp. 25
Figura 10. InputRegisterApp em execução no teste de pose 26
Figura 11. Estrutura do KinectUnity.....26
Figura 13. Trilha do Morro da Cruz em Itajaí. 28
Figura 14. Imagem de interação atual no projeto da trilha do morro 29
Figura 15. Gesto estático simbolizando OK. 32
Figura 16. Gesto Inato.....33
Figura 17. Gesto Aprendido 33
Figura 18. Virtual Foot DPad. 34
Figura 19. Diagrama de classes.....39
Figura 20. Interação com a interface natural 41
Figura 21. Configuração dos comandos naturais 42
Figura 22. Tela de tutorial.....43
Figura 23. Erro no InputRegisterApp 46
Figura 24. Log do KinectServer 49
Figura 25. Análise de performance e monitor de recursos do Windows.....51
Figura 26 - Gráfico Frequência da Experiência 54
Figura 27. Casos de uso menu 70
Figura 28. Casos de uso ambiente.....71
Figura 29. Log do KinectServer 73
Figura 30. Debug no console do Unity 74
Figura 31. Menu InputRegisterApp.....75
Figura 32. Pose reconhecida no InputRegisterApp 76
Figura 33. Configurações do InputRegisterApp.....76
Figura 34. Criado um novo comando de voz 76
Figura 35. KinectServer.....77
Figura 36. InputRegisterApp.....80
Figura 37. InputRegisterApp.....80
Figura 38. InputRegisterApp.....81
Figura 39. KinectServer.....82
Figura 40. Estrutura de pastas do Projeto da Trilha 89

LISTA DE TABELAS

Tabela 1. Interações padrões da ThinkMoto. 34
Tabela 2 - Frequência Experiência.....54
Tabela 3 - Frequência do uso da documentação 55
Tabela 4 - Quantidade de erros.....55
Tabela 5 - Média com desvio padrão 55
Tabela 6 - 1º conjunto de comandos naturais 57
Tabela 7 - 2º conjunto de comandos naturais 57
Tabela 8. Conjunto padrão de comandos naturais.....86
Tabela 9. Conjunto alternativo de comandos naturais 88

LISTA DE QUADROS

Quadro 1. Trecho de script de controle 23
Quadro 2. Exemplo de implementação no Unity 27
Quadro 3. Verificar se lei dos cossenos está ativo.....48
Quadro 4. Verificação se as juntas estão reconhecidas.....50
Quadro 5. Questionário de usabilidade 78
Quadro 6. Iniciar Reconhecimentos 81
Quadro 7. Obter eventos na fila 82
Quadro 8. Reconhecer evento 82
Quadro 9. Questionários de facilidade da biblioteca 83
Quadro 10. Melhoria para reconhecimento de múltiplas poses.....90
Quadro 11. Buscar somente os eventos reconhecidos 91
Quadro 12. Constante do tamanho do buffer 91
Quadro 13. Nome da pose ou comando de voz no log 91
Quadro 14. Botão e fechamento do KinectServer 92

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CG	Computação Gráfica
CLI	*Command Line Interface*
GUI	*Graphical User Interface*
LEA	Laboratório de Educação Ambiental
LIA	Laboratório de Inteligência Artificial
NUI	*Natural User Interface*
SDK	*Software Development Kit*
TTC	Trabalho Técnico-científico de Conclusão de Curso
UML	*Unified Modeling Language*
UNIVALI	Universidade do Vale do Itajaí

SUMÁRIO

1 INTRODUÇÃO ***12***
1.1 PROBLEMATIZAÇÃO *13*
1.1.1 Formulação do Problema *13*
1.1.2 Solução Proposta *13*
1.2 OBJETIVOS *14*
1.2.1 Objetivo Geral *14*
1.2.2 Objetivos Específicos *14*
1.3 METODOLOGIA *15*
1.4 ESTRUTURA DO TRABALHO *16*

2 FUNDAMENTAÇÃO TEÓRICA ***17***
2.1 INTERFACES DE USUÁRIO *17*
2.1.1 Interface CLI *18*
2.1.2 Interface GUI *18*
2.1.3 Interface NUI *19*
2.2 MICROSOFT KINECT *20*
2.2.1 Detalhando o Microsoft Kinect *20*
2.2.2 Microsoft Kinect SDK *21*
2.3 UNITY *22*
2.3.1 Scripts *22*
2.4 API KINECTPROJECT *23*
2.4.1 Estrutura *23*
2.4.2 Restrições *27*
2.4.3 Integração com Unity *27*
2.4.4 Soluções Similares *27*
2.5 PROJETO TRILHA DO MORRO DA CRUZ *28*
2.5.1 Sobre a Trilha do Morro da Cruz *28*
2.5.2 Sobre o Projeto da Trilha *29*
2.5.3 Objetivo do Projeto da Trilha *29*
2.5.4 Implementação *30*
2.5.5 Escolha como estudo de caso *30*
2.6 USABILIDADE E ERGONOMIA *30*
2.6.1 Importância ao projeto em desenvolvimento *31*
2.6.2 Definição de comandos *31*
2.7 TÉCNICAS DE AVALIAÇÃO E VALIDAÇÃO *35*
2.7.1 Pesquisa através da coleta de dados *35*
2.7.2 Testes de software *36*

3 DESENVOLVIMENTO ***38***
3.1 IMPLEMENTAÇÃO DA INTERFACE PARA O PROJETO DA TRILHA *40*
3.1.1 Menu de configuração da interface NUI *42*
3.1.2 Tutorial para aprendizagem dos comandos naturais *42*

4 RESULTADOS ***44***
4.1 AVALIAÇÕES RELACIONADAS AO KINECTPROJECT *44*
4.1.1 Funcionalidade *44*
4.1.2 Desempenho *50*
4.1.3 Facilidade de uso *52*
4.2 QUESTIONÁRIOS *53*
4.2.1 Avaliação da usabilidade na programação com a biblioteca KinectProject *53*
4.2.2 Avaliação da interface natural da Trilha *56*

5 CONCLUSÃO ***58***
5.1 OBJETIVOS ALCANÇADOS *61*
5.2 TRABALHOS FUTUROS *63*
5.3 CONCLUSÕES FINAIS *65*

APÊNDICE A. CENÁRIOS E CASOS DE USO ***70***
A.1 CASOS DE USO *70*
A.2 CENÁRIOS *71*

APÊNDICE B. LOG DO KINECT SERVER ***73***

APÊNDICE C. LOG DO UNITY ***74***

APÊNDICE D. MANUAL DE INSTALAÇÃO E CONFIGURAÇÃO DA TRILHA ***75***

APÊNDICE E. QUESTIONÁRIO DE VALIDAÇÃO DE USABILIDADE ***78***

APÊNDICE F. VALIDAÇÃO DA BIBLIOTECA ***79***

APÊNDICE G. QUESTIONÁRIO DE VALIDAÇÃO DA BIBLIOTECA ***83***

APÊNDICE H. COMANDOS NATURAIS DEFINIDOS ***84***

APÊNDICE I. ESTRUTURA DE PASTAS DO PROJETO DA TRILHA ***89***

APÊNDICE J. TRECHOS DE CÓDIGO DAS CORREÇÕES E ALTERAÇÕES DO KINECTPROJECT ***90***

1 INTRODUÇÃO

Este trabalho reúne conceitos de dois trabalhos realizados por acadêmicos da UNIVALI, sendo estes o KinectProject desenvolvido pelo Bacharel em Ciência da Computação, Gabriel Schade Cardoso e o Projeto Trilha do Morro da Cruz desenvolvido pelo acadêmico em Ciência da Computação, Eduardo A. da Silva.

O KinectProject é uma biblioteca de funções que permite aplicações utilizarem as funções do Microsoft Kinect. A biblioteca captura e reconhece as poses e comandos de voz do usuário obtidos através do Kinect em tempo real durante a execução da aplicação e compara com arquivos de comandos previamente gravados (SCHADE, Gabriel, 2012).

O KinectProject é extensível para diversos tipos de aplicações, sendo somente necessário implementar o cliente, que é o modulo da aplicação que receberá e interpretará os eventos despachados pela biblioteca. A biblioteca já conta com uma implementação para integração ao Unity 3D.

O projeto Trilha do Morro da Cruz implementa uma ferramenta de navegação virtual para a referida trilha (SILVA, Eduardo, 2012). Este projeto utilizou técnicas da área de CG e de Jogos para implementar a modelagem e navegação no ambiente virtual proposto, utilizando a *engine* Unity 3D como ferramenta de desenvolvimento (UNITY, 2013).

O projeto da Trilha consiste em uma modelagem 3D da Trilha do Morro da Cruz, situada em Itajaí, para a criação de um ambiente virtual, conforme mostra a figura 1. Semelhante a um jogo eletrônico, a interação com a trilha virtual é dada através da utilização de mouse e teclado, sendo o teclado para locomoção e o mouse para manipulação da câmera.

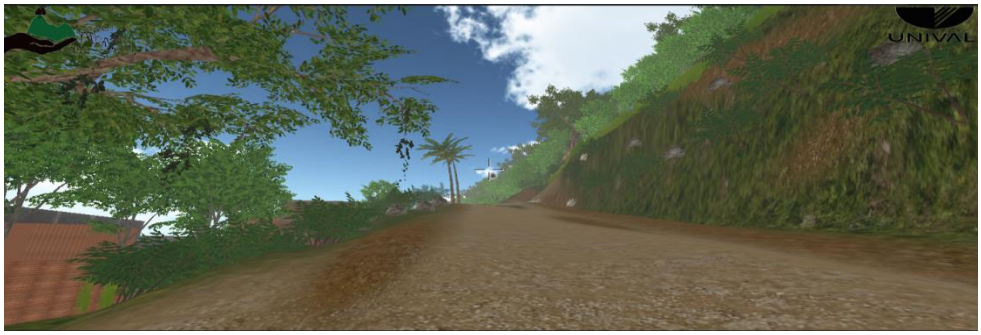


Figura 1. Passeando pelo Projeto da Trilha do Morro da Cruz

O projeto da trilha tem como objetivo ser implantado em diferentes locais da região de Itajaí, inclusive em comunidades que não possuem habilidades e afinidade com tecnologia de informática.

1.1 PROBLEMATIZAÇÃO

1.1.1 Formulação do Problema

No desenvolvimento da biblioteca KinectProject não houve uma avaliação mais detalhada de suas as funcionalidades em uma aplicação real, por este motivo não é possível identificar todas as suas limitações, características de desempenho e de usabilidade.

Levando-se em consideração que a falta de afinidade com tecnologias computacionais nas comunidades as quais o Projeto Trilha do Morro da Cruz será implantado, a forma de interação e a interface atual do projeto da trilha podem tornar um tanto restritivas e frustrantes a experiência de interação com a trilha virtual.

1.1.2 Solução Proposta

Sendo assim, torna-se válido utilizar e validar a biblioteca KinectProject através da implementação de uma interface natural para uma aplicação real. Para tal avaliação foi escolhido o Projeto da Trilha do Morro da Cruz como estudo de caso devido a necessidade de aprimorar a forma de interação atual do mesmo.

Para possibilitar a avaliação e a validação do uso da biblioteca KinectProject no Projeto da Trilha do Morro da Cruz, este trabalho apresenta métricas para facilidade do uso por parte dos desenvolvedores, a aplicabilidade em aplicações não relacionadas a jogos; a análises de performance no reconhecimento de poses e comandos de voz e avaliação de usabilidade na interação com a interface natural por parte dos usuários finais.

Através dos resultados obtidos nas métricas citadas acima se pretende realizar a avaliação e validação da biblioteca KinectProject, tornando-se de conhecimento científico o quanto é confiável e funcional a biblioteca, agregando conhecimento para os futuros desenvolvedores que venham a utilizá-la.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Realizar a validação e avaliação da biblioteca KinectProject através da análise e testes durante e pós implementação da interface natural ao estudo de caso.

1.2.2 Objetivos Específicos

- Estudar e documentar a biblioteca KinectProject desenvolvida pelo Graduado em Computação Gabriel Schade;
- Estudar e documentar a *engine* do Unity 3D;
- Estudar e documentar o Projeto Trilha do Morro da Cruz feita no Unity 3D;
- Estudar e documentar conceitos de ergonomia, usabilidade e acessibilidade relacionados a NUIs;
- Selecionar poses e comandos de voz adequados à interação com o ambiente virtual, baseando-se nos estudos e análises descritos nas etapas anteriores;
- Projetar a interface natural proposta;
- Projetar análises e testes a serem efetuados para validação da ferramenta.

- Produzir a documentação do projeto da interface;
- Implementar a interface natural, através do KinectProject, no projeto da Trilha do Morro da Cruz;
- Implementar um tutorial interativo;
- Executar testes e análises de verificação do funcionamento da interface e ferramenta;
- Produzir um manual de instalação para usuários da nova interface natural do projeto da Trilha Virtual;
- Validar a usabilidade da interface natural através da implantação do projeto no Laboratório de Educação Ambiental – LEA da UNIVALI.
- Documentar todas as etapas envolvidas no desenvolvimento deste TTC na forma de um relatório final e possível artigo científico;

1.3 METODOLOGIA

Este trabalho está dividido em três etapas: (i) fundamentação do projeto, (ii) modelagem da interface natural e dos testes, (iii) considerações gerais e documentação.

Na primeira etapa foram efetuadas pesquisas sobre os conceitos e informações necessários para realização do projeto, nas áreas de usabilidade, interfaces naturais em computação gráfica, APIs e dispositivos relacionados à NUIs. Foram extraídas as informações de artigos científicos, trabalhos de conclusão e documentações online, principalmente da Microsoft, pesquisados utilizando *strings* de busca como: interfaces naturais; NUI; Kinect; usabilidade em NUI; e Unity.

Na segunda etapa foi elaborado a modelagem da implementação da interface natural através de diagramas de classe, casos de uso e cenários com o software Enterprise Architect (SPARX, 2013) e protótipos de tela com o Balsamiq Mockups (BALSAMIQ, 2013), também foi definido quais e como serão realizados os testes da

biblioteca KinectProject com questionários, planos de teste performance e funcionais.

Na terceira etapa foram definidas as considerações gerais em relação ao projeto realizado e toda a documentação do mesmo em um relatório final. Foram registradas desde a apresentação do problema, a solução, testes, resultados e a fundamentação teórica para a realização deste.

1.4 ESTRUTURA DO TRABALHO

Este documento está estruturado em quatro capítulos: (i) Introdução, (ii) Fundamentação Teórica, (iii) Desenvolvimento, (iv) Resultados, e (v) Conclusão.

O Capitulo 1, Introdução, apresenta uma visão geral do trabalho, informando qual a área no qual o projeto está inserido, os objetivos e a metodologia usada.

O Capítulo 2, Fundamentação Teórica, apresenta uma revisão bibliográfica sobre os conceitos, trabalhos similares e informações pertinentes ao projeto como interfaces naturais, usabilidade, Unity, KinectProject, Projeto Trilha Virtual e o Microsoft Kinect.

O Capítulo 3, Desenvolvimento, apresenta a implementação da interface natural ao estudo de caso para realização das validações da biblioteca.

O Capitulo 4, Resultados, apresenta os resultados obtidos e analises dos mesmos a partir dos testes realizados durante o desenvolvimento deste projeto e questionários de avaliação pós implementação.

O Capitulo 5, Conclusão, apresenta as considerações sobre o projeto desenvolvido, como avaliação dos objetivos alcançados, e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Para o desenvolvimento deste trabalho tornou-se necessário realizar uma pesquisa bibliográfica nas diversas áreas relacionadas à esta proposta, são elas: as interfaces de usuários; ambientes virtuais; o dispositivo Microsoft Kinect; o software Unity 3D; a biblioteca KinectProject; o Projeto da Trilha do Morro da Cruz; conceitos de usabilidade e acessibilidade de interfaces NUI; técnicas de projeto e modelagem de interfaces NUI e, por fim, técnicas e metodologias voltadas à avaliação e validação da KinectProject.

A seguir é detalhado cada item informado anteriormente.

2.1 INTERFACES DE USUÁRIO

Interface de usuário é o meio no qual uma pessoa controla um software ou dispositivo de hardware. A qualidade no design destas interfaces proveem uma experiência e interação mais amigável, mais intuitiva (TECHTERMS.COM, 2013). Conforme a figura 2, as três principais interfaces de usuários, são as interfaces CLI, GUI, e NUI.

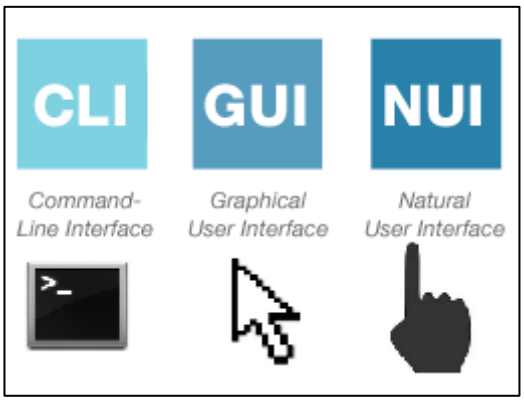


Figura 2. Principais interfaces de usuário

Fonte: Adaptação de GAGE, (2013).

2.1.1 Interface CLI

As interfaces por linhas de comando (*Command Line Interface*) foram as primeiras formas de interação nos mais populares sistemas operacionais nos anos 1970. Softwares que utilizam interfaces CLI necessitam a memorização e conhecimento dos comandos e parâmetros para utilização, sendo considerada para usuários avançados em vários sistemas operacionais (JORDAN, Gustavo, 2013). Na figura 3 é demonstrado um exemplo de interface por linhas de comandos através do *prompt* de comando do Windows.

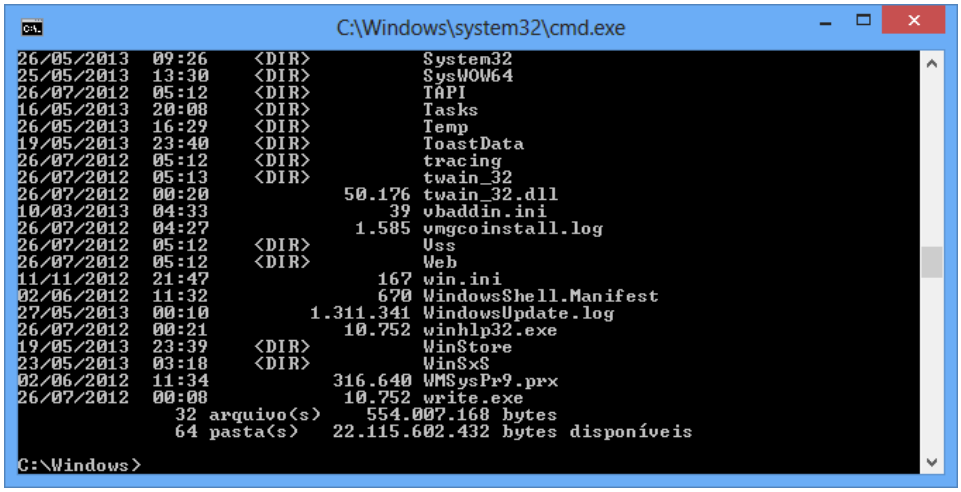


Figura 3. Imagem do prompt de comando do Windows 8

2.1.2 Interface GUI

Interfaces gráficas do usuário, ou GUI (*Graphical User Interface*), é um tipo de interface que permite usuários interagir com equipamentos eletrônicos usando imagens em vez de comandos de texto. Uma interface gráfica possui informações e ações através de ícones ou indicadores visuais, acessíveis normalmente com o uso de mouse e teclado (RAYMOND, Eric, 2013). A figura 4 demonstra o primeiro sistema operacional da Microsoft que se utilizou de interface gráficas.

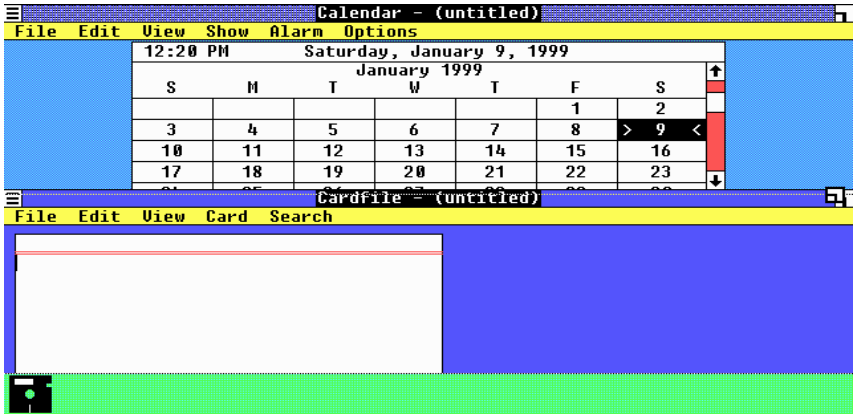

Calendar - (untitled)
File Edit View Show Alarm Options
12:20 PM
Saturday, January 9, 1999
January 1999
S M T W T F S
1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
Cardfile - (untitled)
File Edit View Card Search

Figura 4. Interface gráfica do SO Windows 1.0
Fonte: RAYMOND, Eric (2013).

2.1.3 Interface NUI

O surgimento de interfaces naturais, conhecidas como NUI traz consigo novas formas de interação entre o humano e a máquina. A tecnologia está se tornando mais natural e intuitiva. Pessoas já usam gestos e fala para interagir com dispositivos computacionais, o que torna fácil a aprendizagem na operação de sistemas (MICROSOFT RESEARCH, 2013).

Apesar desse tipo de interface estar surgindo comercialmente nos últimos anos, o estudo de interfaces naturais estima-se que já exista desde a década de 70 com pesquisas realizadas por Steve Mann (CYBORG, 2013).

Interfaces naturais tem como princípio básico a interação com sistemas computacionais através de comandos naturais como gestos, poses e comandos de voz. Gestos são movimentos do corpo, principalmente das mãos, braços e cabeça; já poses são posições que o indivíduo assume e permanece de forma estática (MICHAELIS, 2013).

Um exemplo de ativação de comando utilizando uma interface natural é através do multitoque demonstrado na figura 5, que consiste em uma tela tátil que reconhece múltiplos contatos simultaneamente (ÇETIN, Görkem, 2009). Pode-se também ter comandos mais naturais, como poses ou gestos, como no uso do Microsoft Kinect.

Figura 5. Zoom em uma imagem
Fonte: TURK, TRT (2013).

2.2 MICROSOFT KINECT

O dispositivo de captura de movimentos Kinect desenvolvido pela Microsoft (MICROSOFT KINECT, 2013) é um dispositivo que oferece recursos para a implementação de uma interface natural. Estes recursos são compostos de sensores digitais que capturam os movimentos do usuário e os interpretam transformando estes estímulos em informações no formato digital.

O Kinect utiliza-se de uma tecnologia que permite a interação dos jogadores com jogos sem a necessidade de um controle (*joystick*), diferentemente do Wii (NINTENDO, 2013) e Move da Sony (SONY, 2013) no qual dependem de um controle.

2.2.1 Detalhando o Microsoft Kinect

O sensor Kinect pode ser visto na figura 6. Este sensor possui cerca de 27cm de comprimento horizontal e conta com:

• Emissor de Luz Infravermelha: Emite vários raios infravermelhos para uso da câmera infravermelha;

• Câmera Infravermelha: Através dos raios infravermelhos, utiliza-se a câmera para obter a distância do objeto em relação à câmera;

• Câmera RGB: Captura imagens coloridas em uma resolução de 640x480 e em uma taxa de quadros de 30Hz;

• Microfone embutido: Para captura de comandos de voz;

• Eixo motorizado: Para ajustes do ângulo da câmera.

Figura 6. Imagem do Kinect adaptada
Fonte: SCHADE, Gabriel (2012).

2.2.2 Microsoft Kinect SDK

Em 2011 a empresa Microsoft, liberou um kit de desenvolvimento para o Kinect (SDK), chamado de Kinect for Windows (MICROSOFT KINECT, 2013). O kit é gratuito e atualmente se encontra na versão 1.7. Com ele podemos integrar aplicações Windows com os recursos oferecidos pelo sensor Kinect, utilizando a linguagem C++, C# ou Visual Basic. Nesta mesma época começou a ser vendido o Kinect for Windows, que é especifico para desenvolvedores na plataforma Windows, contendo neste pacote o licenciamento para desenvolvimento. Na figura 7 é possível entender a estrutura do SDK.

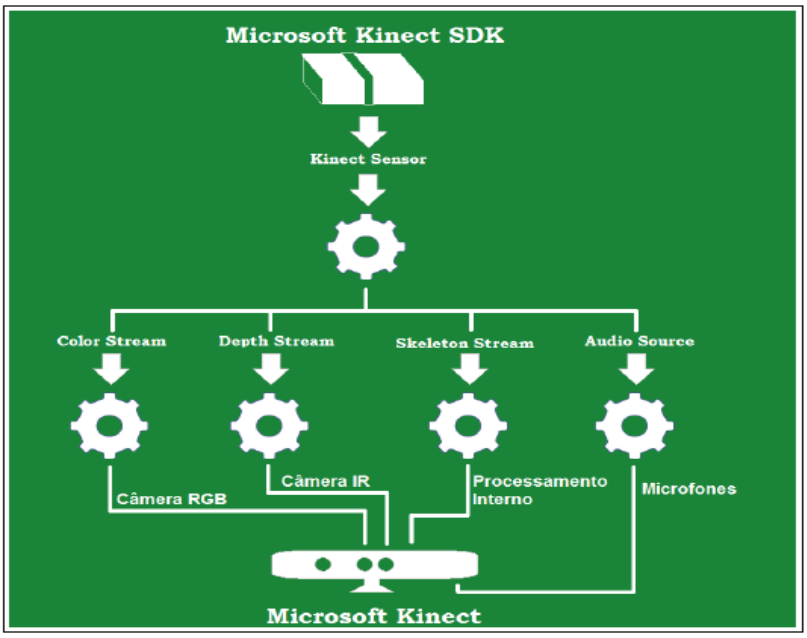


Figura 7. Estrutura do SDK
Fonte: SCHADE, Gabriel (2012).

2.3 UNITY

Unity, também conhecido como Unity 3D, é um motor de jogo 3D, desenvolvido pela Unity Technologies (UNITY, 2013). O Unity oferece ferramentas que auxiliam em todo desenvolvimento de jogos, desde a modelagem à programação. É possível publicar os jogos para diferentes plataformas, como Windows, Mac, Android e Web.

2.3.1 Scripts

Utilizando linguagens, como JavaScript, C# e Boo, é possível codificar scripts no Unity para implementação de animação ao ambiente virtual criado.

A manipulação de entradas é realizada através da criação de um script de *input*, no qual será inserido em um determinado objeto. O Unity possui um script de controle de usuário chamado *FPSWalkerEnchanced* no qual já conta com métodos para locomoção e gravidade, e através deste arquivo é possível manipular as entradas ou como reagir dentro do ambiente (UNITY, 2013).

No quadro 1 está um exemplo de script para o Unity escrito na linguagem C#. Os scripts são comportamentos associados a um objeto da cena, por este motivo o código deste exemplo herda a classe *MonoBehaviour*. O método Update é chamado

a cada frame, sendo assim este é o ponto no código do script onde os desenvolvedores inserem o comportamento desejado. Neste trecho de script, o objeto associado a ele, irá se locomover na posição x 5 pixels a cada frame.

```
01 using UnityEngine;
02
03 public class Teste : MonoBehaviour {
04
05    void Update () {
06          transform.Translate(5f, 0f, 0f);
07    }
08 }
```

Quadro 1. Trecho de script de controle

2.4 API KINECTPROJECT

Com a liberação do Kinect SDK é possível integrar aplicações Windows com os recursos oferecidos pelo sensor Kinect, utilizando a linguagem C++, C# ou Visual Basic. Utilizando este SKD, o Bacharel em Ciências da Computação pela UNIVALI, Gabriel Schade Cardoso, propôs em seu Trabalho de Conclusão de Curso, a biblioteca KinectProject (SCHADE, Gabriel, 2012) que encapsula a SDK do Microsoft Kinect e permite a integração com softwares de desenvolvimento, como o Unity 3D. Através de criação de scripts é possível usar os métodos da KinectProject que permitem acesso às funções do Kinect. Igualmente importante, o KinectProject também oferece a funcionalidade de criação e reconhecimento de poses e comandos de voz capturados através do Kinect, permitindo posterior associação destes à comandos específicos de aplicações desenvolvidas através do Unity 3D.

2.4.1 Estrutura

Para implementação da biblioteca KinectProject foi adotado a filosofia de *dispatcher-listener*, no qual uma aplicação fica responsável por despachar eventos, na qual insere estes eventos em uma fila, como o evento reconhecimento de uma pose por exemplo, e a aplicação final obtém estes eventos da fila (SCHADE, Gabriel, 2013). A biblioteca foi dividida nas seguintes DLLs e aplicações:

KinectScan: DLL responsável por acessar as funções do sensor diretamente através do Microsoft Kinect SDK. Esta DLL é responsável por despachar os eventos reconhecidos, sendo eles: imagem da câmera RGB; informações do esqueleto do usuário; detecção de gestos pré-definidos; detecção de poses e comandos de voz; e

alteração do estado do sensor. E também a funcionalidade de alteração do estado dos eventos descritos anteriormente, como exemplo, iniciar, pausar e parar o evento de reconhecimento de poses.

KinectEvents: DLL que trabalha em conjunto com a KinectScan. Esta DLL conta com as classes que representam os dados do dispositivo Kinect; as classes base da biblioteca como a classe de poses, comandos de voz, entre outras; enumerados para o cada tipo de evento por exemplo, e classes de comunicação com o KinectServer.

KinectServer: Aplicação cliente da DLL KinectScan. Sua finalidade é acessar as funcionalidades da KinectScan de forma que aplicações que não suportem o Framework .NET 4.5 possam interagir como Kinect. Essa funcionalidade é possível pois a aplicação inicia um serviço que pode ser acessado via TCP/IP, ou seja, torna necessário que a aplicação final deve fazer requisições por *socket* para o servidor interagir como o Kinect. A figura 8 demonstra o fluxograma de funcionamento da KinectServer. A DLL KinectScan testa os comando utilizando o SDK da Microsoft, na qual faz comunicação com o hardware, se um comando é reconhecido, um evento da KinectEvents é disparado para o KinectServer, e a aplicação final solicita ao KinectServer os eventos da fila.

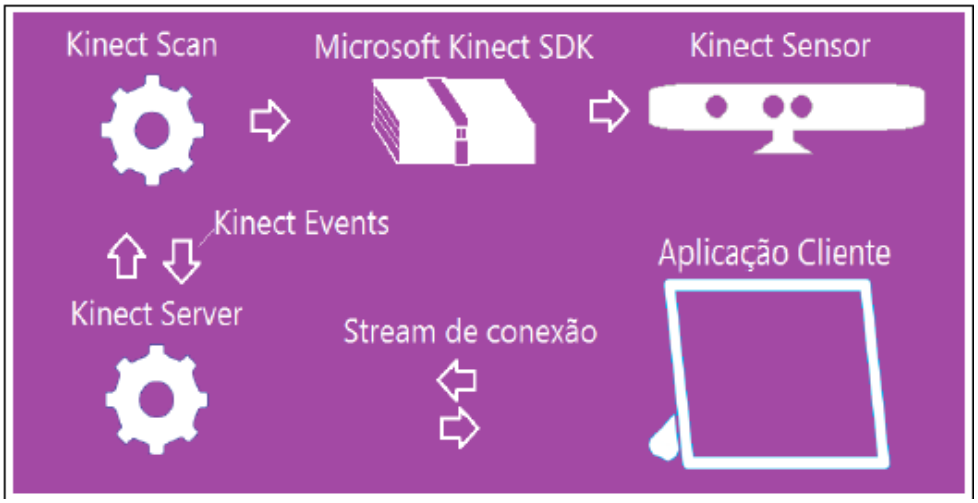


Figura 8. Estrutura do KinectServer
Fonte: SCHADE, Gabriel (2012).

InputRegisterApp: Aplicação desenvolvida para registrar as entradas. Na figura 9 é possível visualizar o fluxograma de funcionamento da ferramenta. Similar ao fluxograma do KinectServer, a DLL KinectScan testa os comandos, que ao ser reconhecido, envia a aplicação final um evento do tipo KinectEvents. Com esta aplicação é possível criar e salvar em arquivo poses e comandos de voz novos. Para a criação de poses, é utilizado um sistema de subposes, margem de erro e tempo de espera. Uma pose pode ser dividida em várias subposes, e cada subpose é um conjunto de três articulações. Para cada subpose é necessário configurar uma margem de erro, para assim facilitar o reconhecimento de uma pose, e que haja o reconhecimento para esqueletos diferentes. Também para finalizar a criação de uma pose é necessário definir o tempo de espera, que significa quanto tempo o usuário deverá manter a pose estática para se tornar como uma pose reconhecida. Para criar um comando de voz somente é necessário escrever a palavra e definir a margem de erro para o reconhecimento. Nesta ferramenta também é possível editar uma pose ou comando de voz já existente como também testar o reconhecimento conforme demonstra a figura 10.

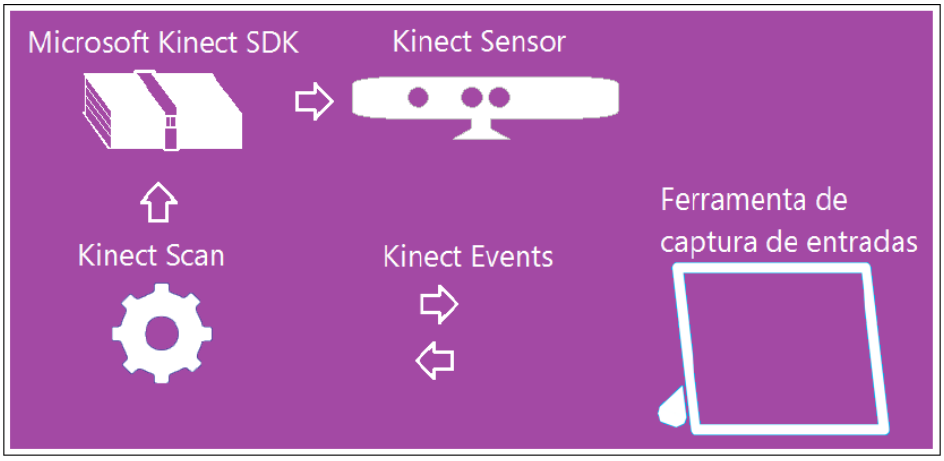


Figura 9. Estrutura do InputRegisterApp
Fonte: SCHADE, Gabriel (2012).

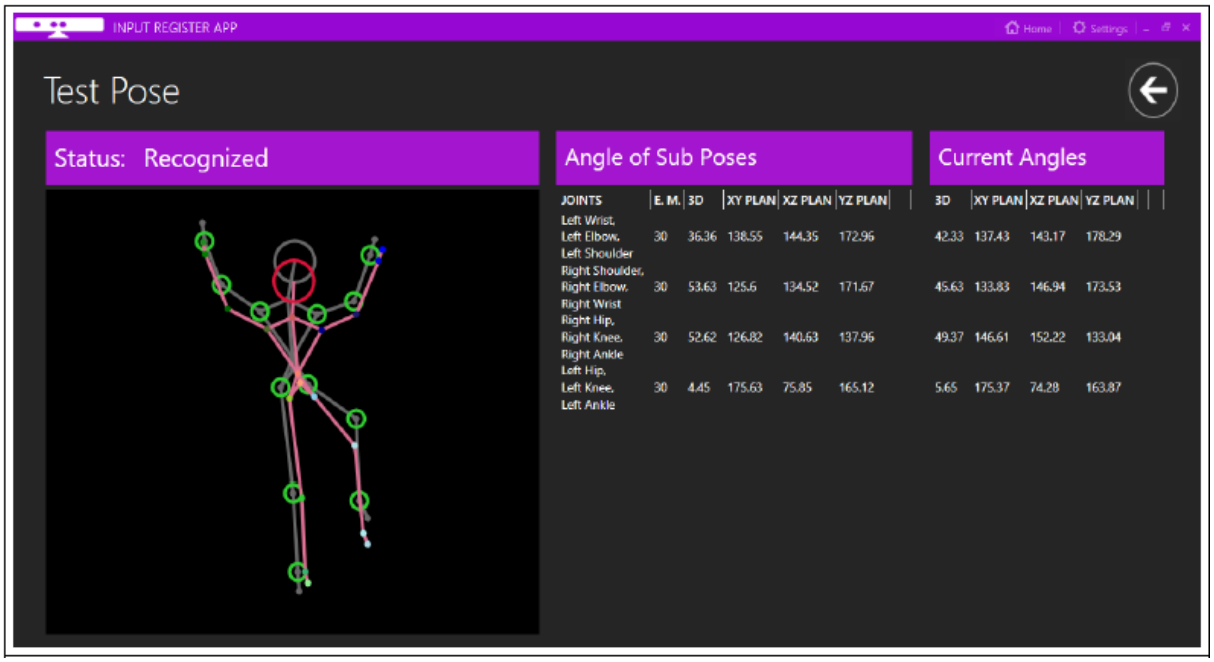

INPUT REGISTER APP
Home
Settings
Test Pose
Status: Recognized
Angle of Sub Poses
Current Angles
JOINTS E. M. 3D XY PLAN XZ PLAN YZ PLAN
Left Wrist, Left Elbow, Left Shoulder 30 36.36 138.55 144.35 172.96
Right Shoulder, Right Elbow, Right Wrist 30 53.63 125.6 134.52 171.67
Right Hip, Right Knee, Right Ankle 30 52.62 126.82 140.63 137.96
Left Hip, Left Knee, Left Ankle 30 4.45 175.63 75.85 165.12
3D XY PLAN XZ PLAN YZ PLAN
42.33 137.43 143.17 178.29
45.63 133.83 146.94 173.53
49.37 146.61 152.22 133.04
5.65 175.37 74.28 163.87

Figura 10. InputRegisterApp em execução no teste de pose
Fonte: SCHADE, Gabriel (2012).

KinectUnity: DLL responsável por consumir os eventos da KinectScan através do servidor KinectServer. Isto ocorre através da utilização da classe KinectServerClient, no qual possui os métodos para enviar requisições e verificar se algum gesto, pose ou comando de voz foi ativo. É possível entender seu funcionamento através da figura 11, no qual segue o mesmo fluxograma do KinectServer.

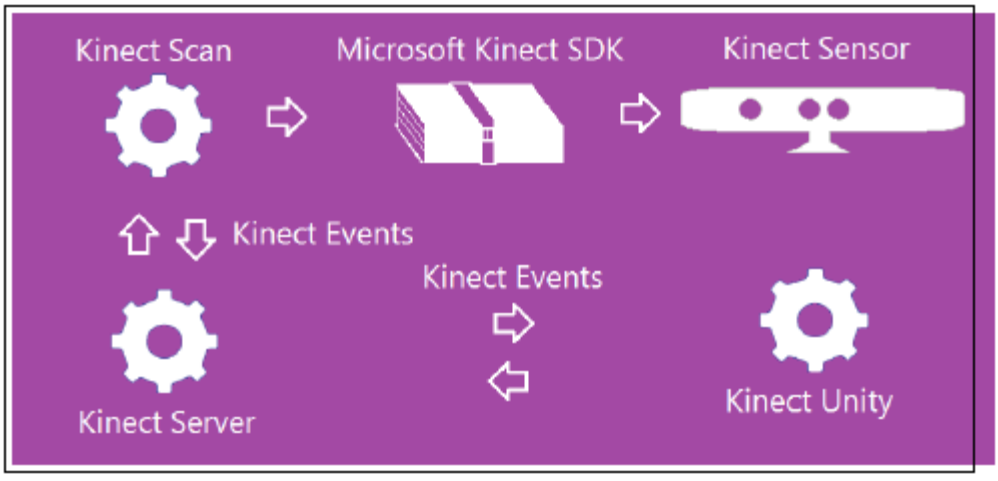


Figura 11. Estrutura do KinectUnity
Fonte: SCHADE, Gabriel (2012).

2.4.2 Restrições

A única restrição inicial segundo o criador da biblioteca está relacionada aos gestos, pois não foi desenvolvida uma ferramenta de gravação e captura de gestos. Sendo assim a biblioteca conta somente com um gesto embutido diretamente na DLL, que é o gesto de mover a mão da esquerda para a direita, ou vice versa, também chamado de *swipe*.

2.4.3 Integração com Unity

A API KinectProject já conta com uma DLL (KinectUnity) para integração com o Unity. É possível através da programação de scripts, acessar e utilizar o Kinect com menos de 10 linhas de códigos.

Na figura 12 é possível visualizar um exemplo de uma implementação. Este exemplo inicia instanciando a classe KinectInput e envia uma requisição para que se inicie o serviço de reconhecimento de comandos de voz no sensor. Durante o método Update no qual é executado a cada frame da aplicação, é enviada uma requisição para que se obtenha todos os eventos reconhecidos. Logo em seguida define-se em uma variável se houve ou não o reconhecimento do comando de voz denominado *Right*.

```
01 void Start(){
02  KinectInput objetoDeInput = KinectInput.Instance;
03  objetoDeInput.SendRequest(new
StartVoiceCommandRequest("Diretório");
04 }
05
06 void Update(){
07  KinectInput.Instance.SendRequest(new TryGetEventsRequest());
08  bool? speechRightMove =
KinectInput.Instance.GetInputVoiceCommand("Right");
09 }
```

Quadro 2. Exemplo de implementação no Unity

2.4.4 Soluções Similares

Foram efetuadas buscas em soluções similares ao KinectProject que suportem o Unity 3D. Uma das soluções encontradas foi Zigfu, que suporta a integração do Kinect com o Unity 3D, JavaScript/HTML 5 e futuramente Flash (ZIGFU, 2013).

Apesar de seu suporte ao Unity, este não possui o reconhecimento de poses. Possui somente o reconhecimento de alguns gestos como swipe, acenar e *push* (empurrar), mas não sendo possível criar novos gestos personalizados.

Caso haja a necessidade de algum tipo de reconhecimento de pose, é necessário implementar manualmente com o uso dos estados obtidos pela biblioteca como posição das juntas.

Maiores detalhes sobre a biblioteca KinectProject podem ser encontradas no trabalho do Gabriel Schade, Biblioteca de Funções para utilização do Kinect em Aplicações GUI e Jogos Eletrônicos (SCHADE, Gabriel, 2013).

2.5 PROJETO TRILHA DO MORRO DA CRUZ

2.5.1 Sobre a Trilha do Morro da Cruz

O Morro da Cruz, situado na cidade de Itajaí, é uma área de grande atrativo turístico na região, surge assim a necessidade de preservação. Atualmente o morro encontra-se em estado degradado. Sendo assim, em 2009, a partir do projeto Restauração da Biodiversidade: Análise do processo de recuperação de uma área do Morro da Cruz (FAPESC, 2009), iniciou-se um processo de recuperação. A figura 13 demonstra o início da trilha acessível pela UNIVALI.

Figura 12. Trilha do Morro da Cruz em Itajaí
Fonte: UNIVALI, Morro da Cruz (2013).

2.5.2 Sobre o Projeto da Trilha

O Projeto da Trilha consiste em um ambiente virtual 3D, desenvolvido na ferramenta Unity, no qual o usuário através do uso de teclado e mouse, se locomove e interage neste ambiente. O projeto foi desenvolvido para acesso somente a área da trilha. Em determinados pontos da trilha, o usuário encontrará indicadores de informações textuais para identificação de um trabalho realizado pelos pesquisadores da Universidade do Vale do Itajaí, assim como informações sobre a biodiversidade ali contida. Existe também a possibilidade de, em alguns pontos, visualizar fotos reais da trilha. A figura 14 mostra um exemplo de interação, no qual um ícone de fotografia é demonstrado no espaço do ambiente e o usuário deve clicar sobre o mesmo para interagir. Ao interagir, a câmera é deslocada e uma imagem real da trilha é subposta no ambiente virtual, sendo necessário o usuário interagir novamente para retornar a trilha.

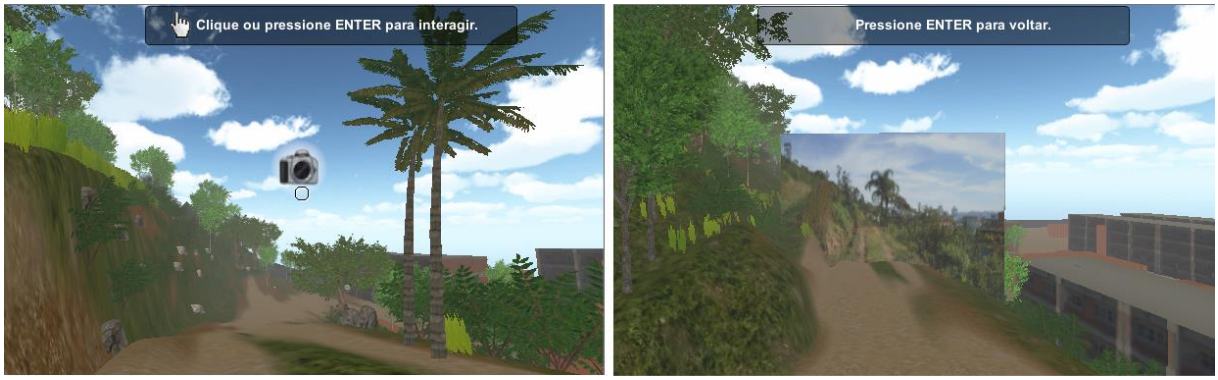


Figura 13. Imagem de interação atual no projeto da trilha do morro

2.5.3 Objetivo do Projeto da Trilha

O Projeto da Trilha do Morro da Cruz tem como objetivo, a disseminação do projeto sendo realizado pelo grupo de pesquisadores de forma digital, tornando desta forma o conteúdo mais acessível e interessante ao usuário final. Os usuários finais terão acesso ao sistema através de centros de integração e nos laboratórios LEA.

2.5.4 Implementação

A implementação do projeto foi dividida em scripts específicos para cada função, como por exemplo, acessar o menu e interagir com uma foto.

Através do objeto denominado Personagem, é possível efetuar a locomoção e a movimentação da câmera. A locomoção é efetuada através da adição do componente *FPSWalkerEnhanced*, o qual é script incluso no Unity que já vem implementado todas as funções para andar ou correr. Já para a movimentação da câmera, é utilizado outro script, o *MouseLook* também incluso no Unity. As mesmas funcionalidades podem ser adaptadas para a implementação da interface natural.

Cada interação visual, textual e menu, representa um script inserido em um componente no Unity. Estes scripts não são inclusos no Unity, foram implementados durante o desenvolvimento do Projeto da Trilha e serão pouco modificados para a adaptação da nova interface.

Segundo SILVA (2013), não foi desenvolvido um projeto de modelagem conceitual da trilha virtual, impossibilitando conhecer o que será ainda desenvolvido no projeto.

2.5.5 Escolha como estudo de caso

O Projeto da Trilha do Morro da Cruz foi escolhido como estudo de caso para este trabalho devido primeiramente por ter sido desenvolvido em Unity, que é por onde será validada a biblioteca KinectProject. Também devido a possibilidade de modificar a forma de interação do projeto da trilha, assim tornando-a mais intuitiva.

Com esta nova forma de interação, pretende-se torna-lo mais atrativo ao usuário final, e assim mais adequado ao objetivo do Projeto da Trilha. Com testes de usabilidade poderá ser verificada a satisfação do usuário final com a nova interface.

2.6 USABILIDADE E ERGONOMIA

Ergonomia é a disciplina relacionada com a interação entre usuários e instrumentos a fim de otimizar o bem estar humano (IEA, 2013). Enquanto usabilidade está relacionada a facilidade de uso.

2.6.1 Importância ao projeto em desenvolvimento

De acordo com princípios ergonômicos, deve-se levar em consideração tanto o componente físico como psíquico dos usuários de um software sendo projetado para que se possa definir as técnicas de NUI adequadas à proposta NUI de interação (SZNELWAR, 2009).

Em um projeto de um software, a interface deve ser flexível visando atender a usuários com diferentes necessidades ergonômicas e de acessibilidade (NORMAN, 2012). No caso de softwares que utilizam conceitos de NUI, deve existir uma preocupação no momento da definição dos gestos, poses e comandos de voz que serão associados às ações. Tais gestos, poses e comandos de voz devem ser escolhidos de forma a tornar mais confortável e intuitiva a interação com o software.

A escolha dos gestos e poses pode influenciar de forma positiva ou negativa em questões de usabilidade e ergonomia dos softwares (NORMAN, 2012). Um exemplo de influência positiva do uso de NUI em relação à usabilidade é a facilidade de compreensão por parte de usuários que não possuem habilidades com tecnologias da informação, além de se tornar um atrativo para o software por possuir uma interface inovadora. E um exemplo onde a utilização dos conceitos de NUI influencia negativamente a usabilidade e ergonomia de um software é a não padronização do conjunto de gestos associados a uma determinada ação, como ocorre nos celulares de diferentes sistemas, com a necessidade de memorização de cada comando (NORMAN, 2012).

Neste trabalho técnico-cientifico, a partir do critério de flexibilidade, é apresentado um menu de configuração, no qual será possível o usuário determinar os comandos naturais a serem utilizados, assim tornando o ambiente ainda mais acessível.

2.6.2 Definição de comandos

Segundo Bill Buxton (2012), a definição dos comandos a serem realizados da melhor forma possível, não depende da pose mais natural, do gesto mais natural ou comando de voz, e sim do conjunto destes, que contribuem para tornar a interface com o sistema mais intuitiva.

A própria Microsoft fornece aos desenvolvedores algumas dicas para desenvolver uma interface natural intuitiva. Cada método de entrada é melhor em alguma coisa e pior em outra. Para início, a entrada deve ser natural e apropriada para o tipo de uso. Caso a interface natural seja implementada em uma aplicação frequentemente usada, como em um escritório, a curva de aprendizagem pode ser maior, e assim possuindo funcionalidades mais ricas. Já em aplicações que raramente são utilizadas, a curva de aprendizagem deve ser mais suave, permitindo que o usuário aprenda no menor tempo possível (MICROSOFT, HIG, 2013).

Resumindo a escolha de uma pose ou gesto de interação deve ser:

- Ergonomicamente confortável;
- Precisa, confiável e rápida; e
- De fácil entendimento, devendo ensinar o usuário qualquer novo tipo de interação.

Um importante detalhe a ser levado em conta na escolha dos gestos é o significado simbólico em diferentes culturas, o símbolo OK visto na figura 15, pode ter outro significado em outra cultura.

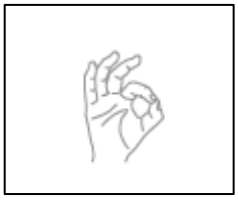


Figura 14. Gesto estático simbolizando OK
Fonte: MICROSOFT, HIG (2013).

De acordo com o HIG (Microsoft HIG, 2013) existem depois tipos de gestos e poses em questões de aprendizagem: gestos inatos e gestos aprendidos. Gestos inatos são aqueles que o usuário conhece intuitivamente, baseado no entendimento do mundo como, por exemplo, apontar para mirar conforme mostra a imagem 16.

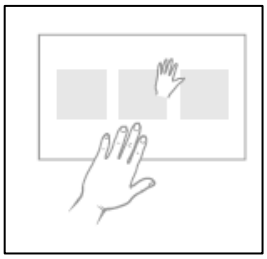


Figura 15. Gesto Inato
Fonte: MICROSOFT, HIG (2013).

Já gestos aprendidos são aqueles no qual é necessário ensinar o usuário a interagir, como, por exemplo, fazer determinada pose para cancelar uma ação ou um gesto como na figura 17.

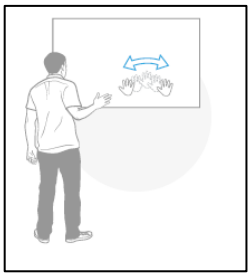


Figura 16. Gesto Aprendido
Fonte: MICROSOFT, HIG (2013).

Dentro destes gestos inatos e aprendidos, estão os gestos estáticos, que representam uma pose estática no qual o sistema reconhece, dinâmicos, que são movimentos com *feedback* continuo como arrastar um objeto, e contínuos que são obter os movimentos em tempo real e transformar em ações continuas. O foco deste projeto será sobre gestos estáticos ou também conhecidos por poses, no qual a biblioteca KinectProject inclui uma ferramenta para gravação destes tipos de gestos.

Como o projeto trata de uma interface natural para a interação em um ambiente virtual e não um jogo, não se vê a necessidade de comandos desafiantes, e sim comandos simples e não cansativos, pois se torna inviável, por exemplo, o usuário se locomover com o braço levantado.

Existem diversas formas de interação como a proposta por Beckhaus, Blom and Haringer (BECKHAUS, 2005) chamada *Virtual Foot DPad + Body Turn.* Consiste basicamente em um direcional semelhante a um *dance pad* só que virtual. O pé a frente indica que se deve locomover em frente e a direção dos ombros indica o movimento da câmera naquela direção. A figura 18 demonstra esta interação.

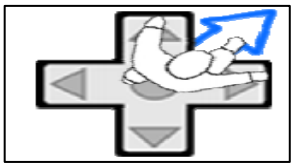


Figura 17. Virtual Foot DPad
Fonte: DAM, Peter; LOAIZA, Manuel; SOARES, Luciano; RAPOSO, Alberto (2012).

Existem outras formas de interação, e algumas empresas como a Think Moto, desenvolvedora de softwares determinou um padrão para interações naturais conforme demostrado na tabela 1 (THINK MOTO, 2013).

	Acenar para mover objetos, rolar ou ativar
	Acenar com a cabeça para cima ou baixo para confirmar ou rolar
	Inclinar-se para direita ou esquerda para mover objeto
	Juntar as mãos para reduzir objeto ou aumentar zoom
	Afastar as mãos para aumentar objeto ou reduzir o zoom
	Mãos no quadril para pausar
	Polegar para cima para confirmar a seleção
	Utilizar o braço para mover um cursor

Tabela 1. Interações padrões da ThinkMoto
Fonte: THINK MOTO (2013).

2.7 TÉCNICAS DE AVALIAÇÃO E VALIDAÇÃO

Para a avaliação e validação da biblioteca KinectProject é necessário conhecimento em realizações de pesquisas com usuários para coletar dados; testes de funcionamento de uma aplicação; e técnicas que são utilizadas para avaliar a usabilidade tanto da interface quanto da biblioteca.

Nas próximas subseções algumas destas técnicas são descritas com mais detalhes.

2.7.1 Pesquisa através da coleta de dados

Para realizar a avaliação deste projeto, se faz necessário realizar pesquisas com usuários para determinar fatores importantes, como a facilidade no uso de uma função disponível no KinectProject por exemplo. Pesquisas descritivas possuem características como facilitar o levantamento de opiniões e atitudes de uma população a respeito de uma situação.

Existem diferentes técnicas para a coleta de dados, como a observação, entrevista e questionários. Para pesquisas descritivas é comum utilizar-se de questionários para coleta de dados (UNIVALI, 2011).

Questionários são utilizados quando há a necessidade de descrever as características de um grupo social e medir algumas de suas variáveis. Tem como vantagens a possibilidade de coletar dados de um grande número de pessoas, assim como a localização ou acesso ao informante não é um fator para realização. Por outro lado, possui desvantagens como um número restrito de questões e baixas taxas de devolução.

Para realização destes questionários, basicamente existem dois tipos de questões, sendo estas abertas e fechadas. Questões abertas são aquelas que dão liberdade ao respondente formular uma reposta, já as fechadas são aquelas que as respostas já são pré-formuladas e cabe ao respondente indicar qual mais se aproxima.

As perguntas fechadas aceitam várias formas de formulação, como por exemplo: sim ou não; escalas de um a dez; e múltipla escolha.

A disposição das perguntas devem seguir alguns critérios como: perguntas fáceis no início, seguidas por perguntas mais especificas com uma complexidade maior e encerrar com perguntas sociodemográficas (idade, sexo, etc.) (UNIVALI, 2011).

Utilizando o software SPSS da IBM (IBM, 2013), que trata-se de uma ferramenta de apoio para análises estatísticas, foi gerado os gráficos e cálculos estatísticos como média, frequência e desvio padrão. Neste software os dados são inseridos, e determina-se o tipo de saída que deseja-se obter.

Através de questionários foram coletados os dados necessários para analisar e obter os resultados das avaliações realizadas neste projeto.

2.7.2 Testes de software

Existem diversos tipos de testes de software, destacando-se dentre eles: testes funcionais; testes não funcionais; testes estruturais e testes de regressão. Para o escopo deste projeto serão abordados os testes funcionais e testes não-funcionais. (GALEOTE, 2013)

Testes funcionais ou também chamados de caixa-preta são aqueles orientados às funções do software, considerando somente o comportamento externo no software, e não a lógica ou estrutura do código. São fornecidas entradas, o teste é executado e compara-se o resultado obtido com o previamente conhecido. Esses testes são baseados em casos de usos ou cenários. (GALEOTE, 2013).

Testes não-funcionais verificam a forma que o sistema trabalha em questões de desempenho, carga ou estresse, usabilidade, portabilidade, entre outros (GALEOTE, 2013).

Uma técnica de avaliação de desempenho consiste em fornecer várias entradas de tamanhos diferentes ao sistema e avalia-se o tempo de resposta para uma determinada funcionalidade onde, normalmente, utiliza-se a medida de milissegundo. O tempo de resposta varia de função para função e de sistema para sistema (REIS, Eberton, 2013).

Na avaliação da usabilidade, pode-se citar quatro técnicas para avaliação de usabilidade (SANTOS, Robson, 2010):

- Inspeção: Utiliza-se de listas de verificação, ou recomendações por *guidelines* para avaliar a usabilidade de uma interface.
- Observação: Através da observação dos usuários do software, registra-se os comportamentos e sua satisfação ou frustação com algum determinado ponto.
- Inquirição: Avalia-se através de questionários, entrevistas ou escalas de avaliação.
- Experimentação: Cria-se uma situação ou caso de teste para que se compare a situação real com a simulada.

Neste trabalho foi utilizada a técnica de inquirição, utilizando-se de questionários, e também a observação para obter informações no qual nos questionários não foram informadas.

3 DESENVOLVIMENTO

Para o desenvolvimento deste projeto, algumas funções foram adicionadas ao Projeto da Trilha do Morro da Cruz de forma a adaptar-se a interface natural, mas sem perder ou reduzir funcionalidades existentes.

Dentro destas funcionalidades está um tutorial para aprendizagem dos comandos NUI; a possibilidade de modificação das entradas para interação dentro da opção de configuração e um *Picture-in-Picture* com o esqueleto do usuário sendo reconhecido em tempo real para *feedback* ao usuário.

O tutorial sobre a interface NUI, acessível através do menu, é um adicional para os usuários iniciais entenderem como interagir com o ambiente. Através deste, será informado aos usuários todas as poses, gestos ou comandos de voz disponíveis para cada tipo de interação com a trilha virtual. Caso nenhuma modificação seja efetuada nas entradas de controle de interação, os comandos a serem exibidos serão os do conjunto padrão, conforme definido no apêndice H.

Para implementação da interface natural, algumas alterações nos requisitos do projeto da trilha foram necessários:

Requisitos funcionais:

- RF01: O sistema deve permitir ao usuário aprender e testar os comandos de interação através de um tutorial
- RF02: O sistema deve permitir a alteração das poses e comandos de voz de cada interação natural

Requisitos não-funcionais:

- RNF01: Utilização do dispositivo Kinect para interação do usuário
- RNF02: Utilização da biblioteca de integração do Unity com o Kinect KinectProject

- RNF03: Utilização de poses e comandos voz definidos através de trabalhos similares e requisitos de usabilidade

Para a codificação da interface, o projeto foi dividido em quatro classes de interação e uma classe principal, conforme figura 19. A classe principal, GlobalInput é a classe que faz a comunicação com a biblioteca KinectProject, iniciando os eventos, obtendo-os da fila e verificando se algum comando foi reconhecido. As outras classes são responsáveis por obter os comandos reconhecidos da classe principal e tratar devidamente com ações ao Projeto da Trilha.

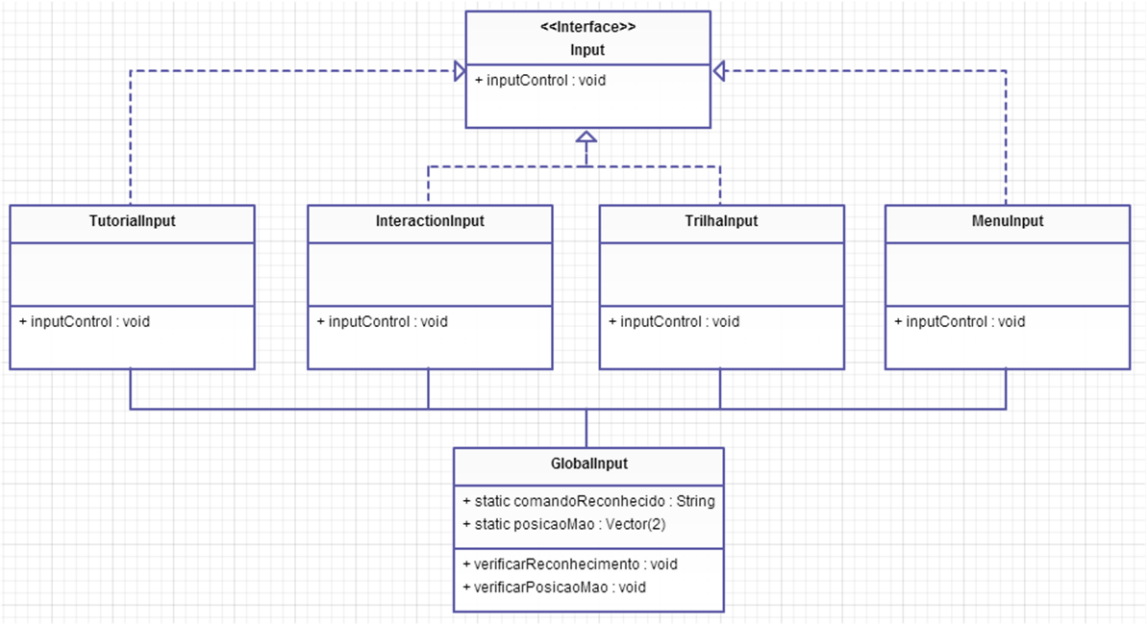


Figura 18. Diagrama de classes

Os cenários e casos de usos definidos ao início deste projeto podem ser consultados no apêndice A.

Como estudado na sessão 2.4, o reconhecimento de poses no KinectProject se dá por um conjunto de subposes e uma margem de erro em relação aos ângulos. Para as poses foram definidas uma margem de erro entre 15% e 30%, dependendo da complexidade da pose. Esta margem foi definida durante a criação e testes do reconhecimento de cada pose. Foi utilizando a aplicação InputRegisterApp que todos os comandos utilizados foram criados, editados e testados.

Foram definidos dois conjuntos de comandos naturais e estes foram baseadas no estudo do HIG da Microsoft, apresentado na sessão 2.6, e padrões de

empresas de desenvolvimento de software como a Think Moto (THINK MOTO, 2013). Dessa forma é possível avaliar o nível de aceitação. Para o segundo conjunto, ocorrerá algumas alterações como a locomoção através do uso do braço direito e acionamento de funções com o braço esquerdo. Por exemplo, o braço direito esticado para frente, sinaliza ao personagem para se locomover a frente.

Todos os testes de validação foram feitos baseados na implementação das funcionalidades ao projeto da trilha virtual, tirando os testes de mudança do ângulo de visão do Kinect e obter as informações da câmera RGB que não seriam utilizados para alguma funcionalidade, somente para validação.

3.1 IMPLEMENTAÇÃO DA INTERFACE PARA O PROJETO DA TRILHA

Conforme descrito na seção 3, o projeto foi dividido em quatro classes, sendo uma para cada seção do ambiente virtual (menu, trilha, interação, tutorial), e uma principal na qual é efetuada a comunicação/integração com a biblioteca KinectProject.

A classe principal de integração foi desenvolvida em C# e inserida na pasta *Standard Assets*, para que assim o Unity o compile antes das outras classes. Essa modificação foi necessária para que as outras classes, que foram desenvolvidas em Unity Script, possam acessar as informações contidas na classe principal denominada GlobalInput.

As classes de cada seção foram desenvolvidas em Unity Script para melhor compatibilidade com os outros scripts desenvolvidos para a Trilha do Morro da Cruz. Estas classes acessam a variável que possui o mapeamento dos comandos que estão sendo reconhecidos. Quando um determinado comando é reconhecido, acessa-se o objeto necessário e altera-se suas propriedades.

Para a alteração das propriedades dos scripts dos objetos da trilha, foi necessário alterar as variáveis de controle utilizadas, como a variável da posição atual da aba de uma interação textual. Estas variáveis foram alteradas de permissão privada para permissão pública, assim obtendo acesso a partir das classes de integração.

Ao iniciar a aplicação, os eventos de reconhecimento de pose e gestos são iniciados e pausados. Somente após receber o evento de comando de voz “Kinect”, os eventos de poses e gestos são retomados e todos comandos são testados no *script*. Caso o comando de voz “b*ye*” seja reconhecido, os eventos de pose e gestos são pausados novamente e os comandos não são mais testados. Essa função foi implementada afim de ser possível ativar e desativar a interface natural evitando reconhecimentos indesejados. Para melhor identificação quando a interface natural é ativada ou desativada, um som de bip é tocado.

A figura 20 demonstra uma interação sendo realizada após a implementação da interface. O lado A demonstra o Kinect, a aplicação KinectServer executando e o Projeto da Trilha em execução. O lado B é a trilha virtual antes da execução de algum comando natural. No lado C o usuário realiza um comando natural na qual aciona a interação textual visualizada no lado D.

Figura 19. Interação com a interface natural

3.1.1 Menu de configuração da interface NUI

É possível através de um menu alterar as configurações da interface natural. Conforme a imagem 21 mostra, todos os comandos estão listados e pode-se ativar uma pose, um comando de voz e/ou um gesto. Nos campos de texto deve-se inserir o nome do comando a ser reconhecido. Novas poses e comandos de voz deverão ser criados pela aplicação InputRegisterApp e inseridos em suas respectivas pastas na pasta “data” da Trilha do Morro da Cruz. A estrutura das pastas pode ser visualizada no apêndice I.

Figura 20. Configuração dos comandos naturais

3.1.2 Tutorial para aprendizagem dos comandos naturais

Um tutorial foi implementado para permitir que todos os comandos naturais existentes sejam apresentados, testados e verificados pelo usuário.

O tutorial apresenta uma interface gráfica que informa qual comando está sendo testado. Se este comando possuir um comando de voz, a palavra ou frase é exibida na tela; se possuir gesto, o gesto é descrito através de um texto; e se possuir uma pose, o esqueleto gravado no arquivo da pose é exibido ao usuário.

Conforme a figura 22, quando o comando é reconhecido, uma mensagem informa que o comando já foi reconhecido e o usuário deverá falar “OK” para ir para

o próximo comando. Além disto, quando um comando é reconhecido, a frase “Comando reconhecido” também aparece e se possuir uma pose, o esqueleto fica da cor verde, e após cerca de quatro segundos a frase desaparece e o esqueleto do usuário retornam para a cor normal.

Esta forma de *feedback* de reconhecimento de comandos foi implementada para que o usuário possa efetuar o teste novamente caso tenha realizado o comando sem intenção.

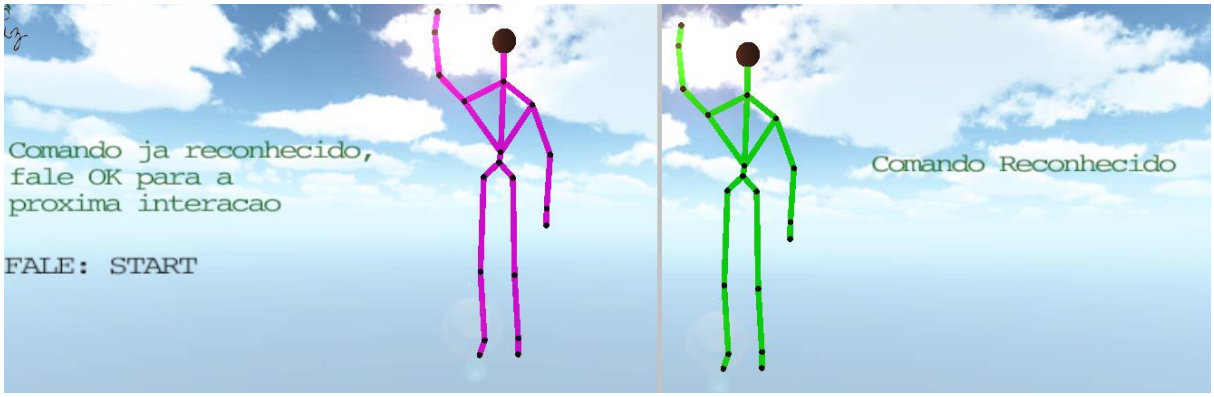


Figura 21. Tela de tutorial

A função de *Picture-In-Picture* do esqueleto do usuário em tempo real teve de ser removida do projeto devido a limitação da biblioteca, conforme explicado em mais detalhes na sessão 4.1 deste trabalho.

4 RESULTADOS

4.1 AVALIAÇÕES RELACIONADAS AO KINECTPROJECT

Nas próximas seções estão descritas todas as avaliações realizadas e suas respectivas análises, juntamente com um levantamento de falhas encontradas. As avaliações foram divididas em funcionalidade, desempenho e facilidade de uso.

4.1.1 Funcionalidade

Durante a implementação da interface para o Projeto da Trilha do Morro da Cruz foram identificados problemas com algumas funcionalidades da KinectProject que impediriam cumprir totalmente os objetivos deste TTC.

Com o aplicativo InputRegisterApp foi possível criar e editar poses e comandos de voz corretamente. Inclusive, as funções de teste destes comandos criados também funcionaram de forma correta. Foram identificados alguns erros não tratados como por exemplo o item falha 2 da seção 4.1.1.1.

Os testes de reconhecimento de gestos e de voz realizados obtiveram sucesso, nenhum problema foi identificado. Os testes foram realizados através da utilização destes comandos naturais para realizar algumas interações como mover a câmera e acionar uma interação textual no projeto da trilha.

A execução dos comandos de iniciar, pausar e parar os eventos de reconhecimento do KinectProject funcionaram corretamente.

Também foi possível alterar o ângulo de visão do Kinect, função descrita na seção 2.2.1, ocorrendo apenas uma erro descrita no item restrição 2 da seção 4.1.1.3.

Foi realizado um teste básico do reconhecimento de poses no início deste projeto através do minijogo Jump Game. Neste teste, foi possível verificar o reconhecimento de poses em ação. Durante os testes da implementação da trilha, percebeu-se que o reconhecimento de poses não estava funcional, as poses não eram reconhecidas normalmente. Foi então identificada a falha 5 da seção 4.1.1.1 falhas e a necessidade da melhoria 2, da seção 4.1.1.2 melhorias.

4.1.1.1 Falhas

Aqui estão listadas as falhas identificadas na biblioteca, sua descrição, análise de correção realizada e se foi possível corrigir. As falhas foram divididas por componente da KinectProject.

I. KinectScan:

Falha 1: Impossibilidade de obter esqueleto do usuário.

Descrição: Quando há tentativa de obter as juntas do esqueleto do usuário em tempo real, uma exceção é levantada. Ao testar em modo *debug* uma exceção de objeto não marcado como serializável é levantada.

Correção: A classe KinectScanUserSkeleton da KinectEvents utiliza uma coleção do tipo Dictionary que não é serializável, então criou-se e substitui-se por uma classe dicionário serializável. Notou-se então que além deste problema, o buffer de comunicação que estava fixo para o usuário final em 1.024 bytes, não era o suficiente para enviar as informações do esqueleto do usuário, então o mesmo foi fixado para 20.000. Com estas alterações a obtenção do esqueleto do usuário funcionou corretamente quando testado, sendo necessário somente adaptar outras aplicações como o InputRegisterApp para utilizar o mesmo dicionário serializável. Seria interessante possibilitar ao usuário final alterar o tamanho do buffer na aplicação, removendo assim a constante fixa no código.

Status: Problema de obtenção do esqueleto do usuário resolvido

II. **InputRegisterApp:**

Falha 2: Impossibilidade de salvar um comando criado no InputRegisterApp em uma pasta sem permissão de alteração do sistema operacional.

Descrição: Ao tentar salvar o arquivo do comando criado no InputRegisterApp em uma pasta que o aplicativo não possui permissão de alteração no sistema operacional, o aplicativo para de responder, conforme pode ser visualizado na figura 23.

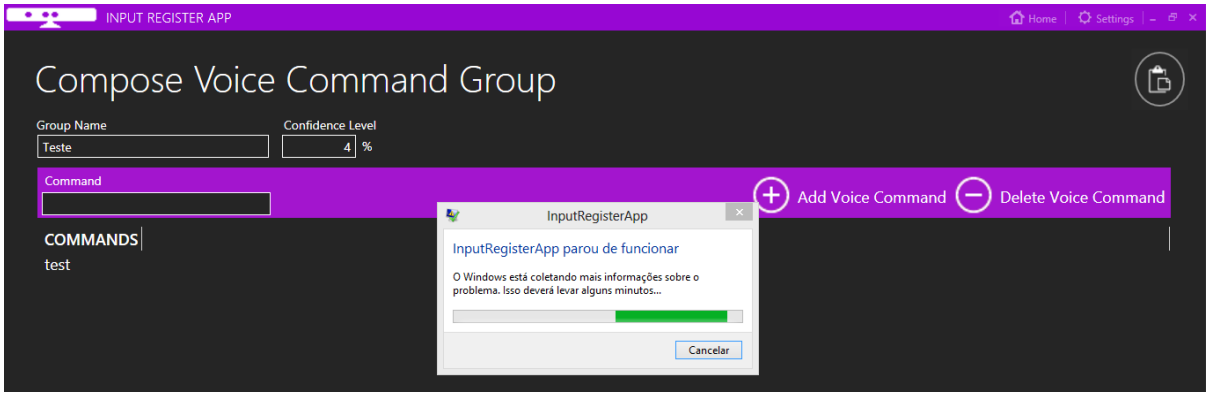


Figura 22. Erro no InputRegisterApp

Correção: Problema identificado como uma exceção levantada e não tratada pela aplicação, porém ainda não corrigido por não pertencer ao escopo deste trabalho e ser uma correção de baixa prioridade para a implementação da interface natural.

Status: Pendente

III. KinectServer:

Falha 3: KinectServer não encerra corretamente quando existem eventos em execução

Descrição: Na aplicação KinectServer, ao fechar a aplicação com eventos rodando, a interface fecha e o processo do KinectServer continua em execução, sendo necessário finaliza-lo pelo gerenciador de tarefas do sistema operacional.

Correção: Foi corrigido o código fazendo com que a aplicação force a finalização dos processos envolvidos.

Status: Problema resolvido.

Falha 4: Botão do KinectServer para ativar/desativar a comunicação com o Kinect com falhas

Descrição: O botão para ativar/desativar a comunicação com o Kinect também está com falhas, o mesmo inicia corretamente, mas ao tentar desativar para parar todos os eventos em execução, nada acontece.

Correção: Foram efetuadas correções no botão para destruir os objetos para que se possa recria-los ao iniciar novamente.

Status: Problema resolvido.

Falha 5: Falha no método de reconhecimento de poses quando executado através do KinectServer

Descrição: Percebeu-se que através do KinectServer, o reconhecimento de poses não funciona como deveria, se torna falho, principalmente quando testado com uma maior diversidade de poses. O problema só ocorre no KinectServer, enquanto o InputRegisterApp o reconhecimento é normal.

Correção: Através da análise do código, comparando a execução do reconhecimento de pose com o KinectServer e com o InputRegisterApp, notou-se que apesar de ambos utilizarem as mesmas classes e funções para reconhecimento de poses, através do KinectServer, o trecho do quadro 2 da classe KinectFunctionsManager não era executado. Foi então identificado que somente para o KinectServer, existia uma variável global, que definia que o uso da lei dos cossenos estava desativa. Após definir a variável como verdadeiro, o cálculo das lei dos cossenos começou a ser utilizado na comparação das poses, tornando o reconhecimento normalizado. Apesar disto, é necessário corrigir o cálculo através do produto escalar por questões de performance.

Status: Problema parcialmente resolvido.

```
01 if (returnValue && useLawOfCosines)
02             {
03                 LawOfCosinesResult lawOfCosinesResult =
MathFunctionsManager.LawOfCosinesCalculation( userSubPose );
04                 returnValue &=
MathFunctionsManager.ComparisonWithErrorMargin(
subPoseForTest.Angle.LawOfCosinesResult.PlanXY,
lawOfCosinesResult.PlanXY, subPoseForTest.AngleMarginError );
05                 returnValue &=
MathFunctionsManager.ComparisonWithErrorMargin(
subPoseForTest.Angle.LawOfCosinesResult.PlanXZ,
lawOfCosinesResult.PlanXZ, subPoseForTest.AngleMarginError );
06                 returnValue &=
MathFunctionsManager.ComparisonWithErrorMargin(
subPoseForTest.Angle.LawOfCosinesResult.PlanYZ,
lawOfCosinesResult.PlanYZ, subPoseForTest.AngleMarginError );
07             }
```

Quadro 3. Verificar se lei dos cossenos está ativo

Falha 6: Não é possível receber as informações da câmera RGB do Kinect

Descrição: Não é possível receber os bytes referentes ao RGB da câmera do Kinect através do KinectServer, uma exceção é levantada constantemente na DLL KinectServer de sem memória e ao tentar obter pelo Unity ocorre um erro na deserialização.

Correção: Problema não identificado, mas com possíveis problemas na transmissão no qual utiliza funções de serialização, e o limite do buffer ser muito pequeno para transmitir a imagem da câmera. Problema ainda não corrigido por não pertencer ao escopo deste trabalho e por ser uma correção de baixa prioridade para a implementação da interface natural.

Status: Pendente.

4.1.1.2 Melhorias

Melhoria 1: Log do KinectServer não identifica qual o nome do comando reconhecido

Descrição: O log do KinectServer identifica corretamente o horário e o tipo do evento reconhecido pelo KinectScan, mas não demonstra o nome do comando.

Alteração: Foi alterado o código para inserir no registro do log, o nome das poses ou comandos de voz reconhecidos, conforme demonstrado na figura 24.

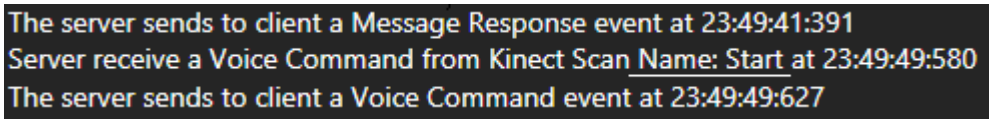

The server sends to client a Message Response event at 23:49:41:391
Server receive a Voice Command from Kinect Scan Name: Start at 23:49:49:580
The server sends to client a Voice Command event at 23:49:49:627

Figura 23. Log do KinectServer

Melhoria 2: Reconhecimento de poses simultâneas.

Descrição: O reconhecimento de poses só permite enviar um evento de pose reconhecida por vez, sendo necessário que nenhuma pose esteja sendo reconhecida para que reconheça uma nova. Por exemplo, enquanto o usuário realizar a pose com o pé à frente, outras poses como um braço esticado, não é reconhecido, somente reconhecerá se o usuário retirar o pé à frente. Isso é necessário para cumprir um requisito de usabilidade produzindo assim várias entradas ao mesmo tempo.

Alteração: Foi alterado o código para que todas as poses continuem sendo verificadas, independente se já existe uma com o status reconhecido.

Melhoria 3: Verificar se as juntas da subpose que irá ser testada estão sendo reconhecida.

Descrição: Os cálculos para reconhecimento de poses são efetuados mesmo que alguma junta da subpose a ser testada não esteja sendo reconhecida.

Alteração: Foi inserido o código do quadro 3 no qual verifica se todas as juntas das subposes do esqueleto do usuário, que serão comparadas com os arquivos de poses, estão sendo reconhecidos.

```
01  if
(!centerJoint.TrackingState.Equals(KinectEvents.Enums.KinectScanJointT
rackingState.Tracked) ||
!auxiliaryJoint1.TrackingState.Equals(KinectEvents.Enums.KinectScanJoi
ntTrackingState.Tracked) ||
!auxiliaryJoint2.TrackingState.Equals(KinectEvents.Enums.KinectScanJoi
ntTrackingState.Tracked))
02              {
03                  return false;
04              }
```

Quadro 4. Verificação se as juntas estão reconhecidas

4.1.1.3 Restrições

Restrição 1: Impossibilidade da obtenção do esqueleto do usuário juntamente com outros eventos.

Descrição: Após a implementação de um Picture-In-Picture do esqueleto do usuário no Projeto da Trilha, o reconhecimento de voz, pose e gestos deixava de funcionar corretamente. Notou-se que os eventos do esqueleto do usuário, eram enviados constantemente a cada ciclo da biblioteca, fazendo que a fila aumentasse consideravelmente e impedindo o reconhecimento de outros eventos. Seria necessário alterar a estrutura da biblioteca, criando uma segunda fila de eventos somente para o reconhecimento do esqueleto do usuário.

Restrição 2: Erros durante a alteração do ângulo de visão do Kinect

Descrição: Quando uma nova solicitação de alteração do ângulo do Kinect é enviada antes de uma anterior ter sido finalizada, uma exceção é levantada, originando do SDK do Kinect.

4.1.2 Desempenho

Utilizando o monitor de recursos do Windows e a análise de performance da IDE Visual Studio 2012, foi possível verificar o consumo dos recursos de hardware. Foram duas avaliações, a primeira da execução do projeto da trilha na sua versão original onde a interação é realizada via teclado e mouse, e a segunda execução da versão final com a interface natural implementada. Na segunda execução, é também avaliado o desempenho do KinectServer.

A figura 25 demonstra do na parte superior o analisador de performance da IDE Visual Studio no qual demonstra resultados de uso da CPU e também da memória sendo utilizada por cada componente da biblioteca. Já na parte inferior demonstra o monitor de recursos do Windows com as informações dos processos em execução. Cada teste é executado duas vezes por um período de 10 minutos.

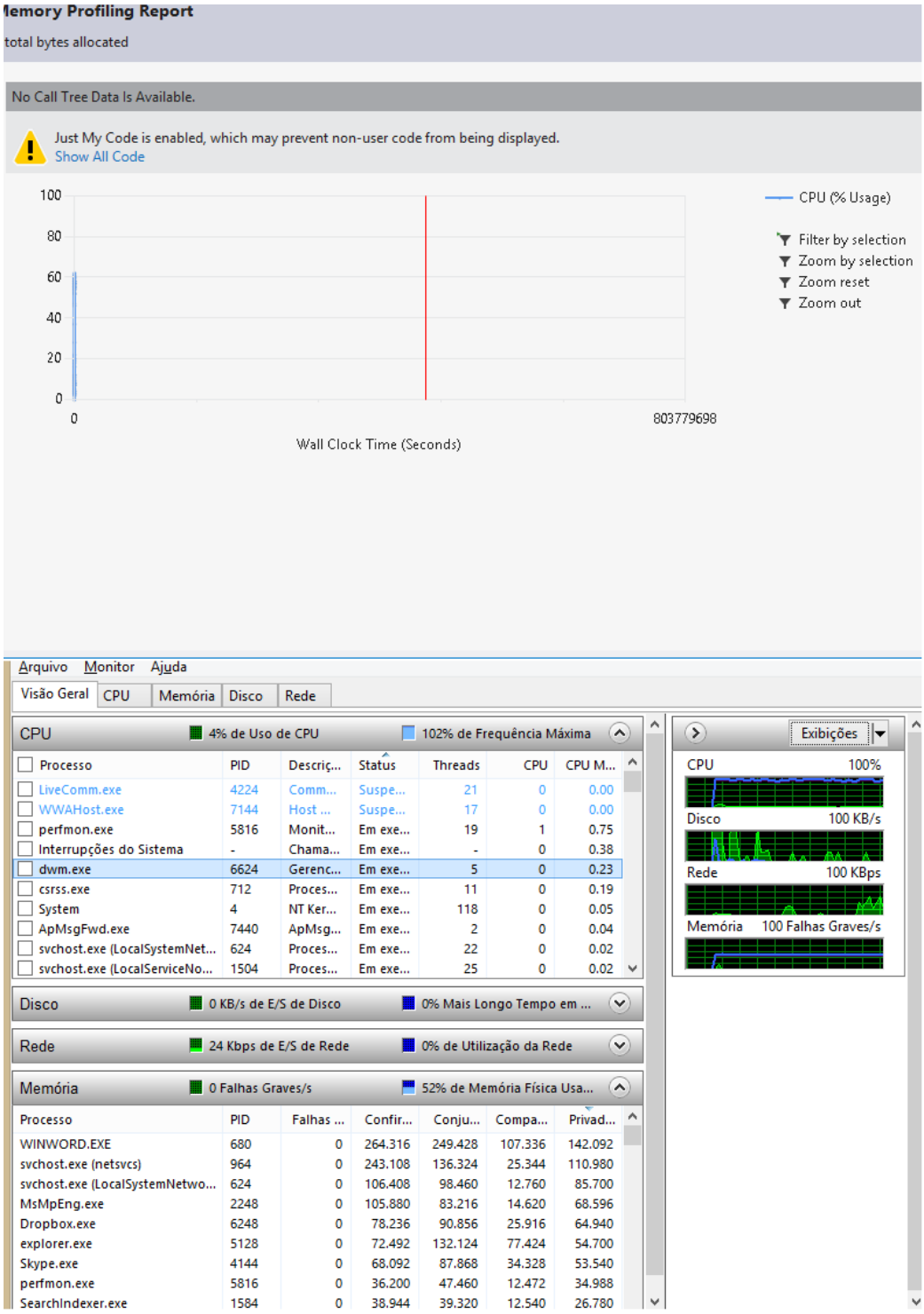

Memory Profiling Report
total bytes allocated
No Call Tree Data Is Available.
Just My Code is enabled, which may prevent non-user code from being displayed.
Show All Code
100
80
60
40
20
0
0
803779698
Wall Clock Time (Seconds)
CPU (% Usage)
Filter by selection
Zoom by selection
Zoom reset
Zoom out
Arquivo Monitor Ajuda
Visão Geral CPU Memória Disco Rede
CPU 4% de Uso de CPU 102% de Frequência Máxima
Processo PID Descriç... Status Threads CPU CPU M...
LiveComm.exe 4224 Comm... Suspe... 21 0 0.00
WWAHost.exe 7144 Host ... Suspe... 17 0 0.00
perfmon.exe 5816 Monit... Em exe... 19 1 0.75
Interrupções do Sistema - Chama... Em exe... - 0 0.38
dwm.exe 6624 Gerenc... Em exe... 5 0 0.23
csrss.exe 712 Proces... Em exe... 11 0 0.19
System 4 NT Ker... Em exe... 118 0 0.05
ApMsgFwd.exe 7440 ApMsg... Em exe... 2 0 0.04
svchost.exe (LocalSystemNet... 624 Proces... Em exe... 22 0 0.02
svchost.exe (LocalServiceNo... 1504 Proces... Em exe... 25 0 0.02
Disco 0 KB/s de E/S de Disco 0% Mais Longo Tempo em ...
Rede 24 Kbps de E/S de Rede 0% de Utilização da Rede
Memória 0 Falhas Graves/s 52% de Memória Física Usa...
Processo PID Falhas ... Confir... Conju... Compa... Privad...
WINWORD.EXE 680 0 264.316 249.428 107.336 142.092
svchost.exe (netsvcs) 964 0 243.108 136.324 25.344 110.980
svchost.exe (LocalSystemNetwo... 624 0 106.408 98.460 12.760 85.700
MsMpEng.exe 2248 0 105.880 83.216 14.620 68.596
Dropbox.exe 6248 0 78.236 90.856 25.916 64.940
explorer.exe 5128 0 72.492 132.124 77.424 54.700
Skype.exe 4144 0 68.092 87.868 34.328 53.540
perfmon.exe 5816 0 36.200 47.460 12.472 34.988
SearchIndexer.exe 1584 0 38.944 39.320 12.540 26.780
Exibições
CPU 100%
Disco 100 KB/s
Rede 100 KBps
Memória 100 Falhas Graves/s

Figura 24. Análise de performance e monitor de recursos do Windows

Os testes foram realizados utilizando as seguintes especificações de hardware: processador Intel Core i5-2410M 2.30GHz; memória DDR3 4GB 1333MHz; placa de vídeo AMD Radeon HD 6630M de 1GB, 128-bit e disco rígido de 7200RPM.

Através dos testes com a utilização da biblioteca foi verificado que o KinectServer aumenta cerca de 200 KB por segundo de memória virtual, e utiliza em média 150 MB da memória de trabalho. Percebeu-se também, que a quantidade de *threads* para o processo do KinectServer, aumenta progressivamente, sendo em média duas *threads* por segundo. Após dez minutos de utilização, nota-se uma queda de desempenho da aplicação, erro o qual pode ser proveniente da quantidade excessivas de threads em execução.

O projeto da trilha com a interface natural implementada, com os *plug-ins* carregados e executando os scripts de reconhecimento, obteve um aumento de 20% de uso da CPU e 15% da memória em comparação com os testes realizados com o projeto da trilha original sem a interface natural.

Falha de desempenho: Alto processamento necessário para testar se os comandos foram reconhecidos:

> **Descrição:** A implementação original do KinectUnity necessitava que o desenvolvedor informasse o nome do comando a ser testado, de forma individual, e era retornado um valor booleano representando se o comando foi reconhecido ou não. Foi então realizada uma alteração na estrutura para que se possa retornar o nome dos comandos que foram reconhecidos, e então mapeados no Unity para cada ação, diminuindo assim consideravelmente o processamento dos scripts.

4.1.3 Facilidade de uso

A biblioteca KinectProject é simples e com poucas linhas de códigos é possível realizar o reconhecimento de comandos naturais. Mas devido à falta de uma documentação mais completa e com exemplos de como executar cada função, torna-se difícil a implementação.

Falha de usabilidade: Documentação incompleta

Descrição: Para efetuar testes na obtenção do esqueleto do usuário, por exemplo, recorrer à documentação não foi suficiente, algumas horas foram gastas em testes até a obtenção correta dos dados. Além da falta de exemplos, não há uma explicação mais detalhada de cada função.

4.2 QUESTIONÁRIOS

4.2.1 Avaliação da usabilidade na programação com a biblioteca KinectProject

O questionário para avaliar a biblioteca KinectProject foi realizada por programadores, e por estudantes do curso de Ciência da Computação da UNIVALI, incluindo a turma de Tópicos Especiais em Computação I do semestre II de 2013, totalizando onze participantes.

Foi solicitado aos participantes que em posse da biblioteca e aplicações do KinectProject, modificassem um minijogo desenvolvido no Unity para utilização de comandos naturais com o Microsoft Kinect. O pacote constava:

- Um simples tutorial de passo a passo do que realizar, no qual consta no apêndice F.
- Documentação da biblioteca KinectProject
- O minijogo Jump Game desenvolvido no Unity
- As aplicações KinectServer e InputRegisterApp
- As biblioteca de integração KinectProject
- Questionário

Os participantes tiveram que responder seu nível de conhecimento em programação; se conseguiu concluir o teste; se recorreu à documentação da biblioteca e sobre a qualidade da mesma; quantos erros ocorreram e uma nota de

zero a dez em relação a facilidade de uso da biblioteca. O questionário pode ser visualizado no apêndice G.

Os resultados obtidos foram divididos em três populações, sendo estas definidas pelo conhecimento ou experiência em programação. Questionários com informações incompletas foram descartados, ao invés de inferir os dados faltantes. No total foram descartados dois questionários.

A realização dos cálculos e geração dos gráficos foram efetuadas pela ferramenta IBM SPSS Statistics (IBM, 2013).

Para este questionário, houveram onze participantes. Destes participantes, a maioria possuía conhecimento básico ou intermediário em programação, somando juntos aproximadamente 82% do total, conforme visualizado na tabela 2 e no gráfico da figura 26.

Nível Experiência	Frequência	%
Básico	4	36,4
Intermediário	5	45,5
Avançado	2	18,2
Total	11	100,0

Tabela 2 - Frequência Experiência

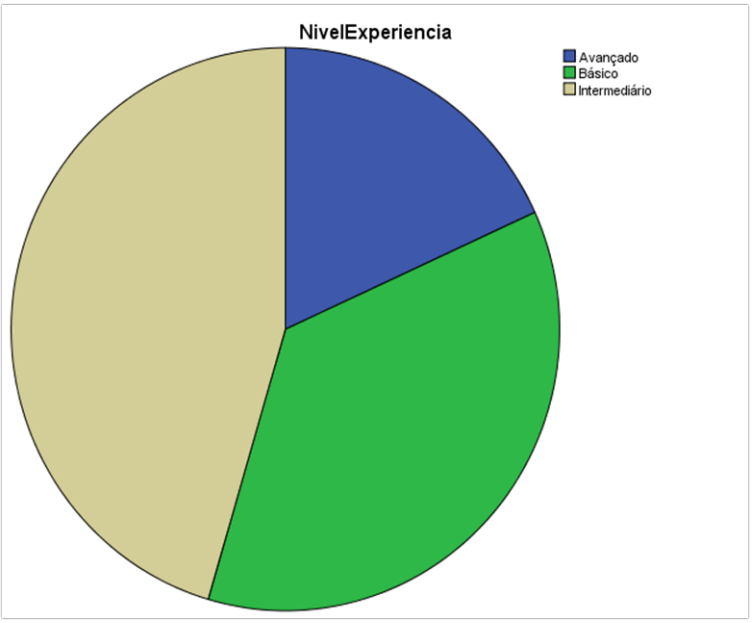


Figura 25 - Gráfico Frequência da Experiência

Destes participantes, 90% conseguiram concluir o teste realizado. E como pode ser visualizado nos quadros 3 e 4, 36% recorreram a documentação e 55% obtiveram um ou mais erros.

Recorreu documentação	%
Sim	36,4
Não	63,6
Total	100,0

Tabela 3 - Frequência do uso da documentação

Erros	%
0	45,5
1	36,4
2	9,1
4	9,0
Total	100,0

Tabela 4 - Quantidade de erros

Dos que recorreram a documentação, 75 % avaliaram o nível da mesma como péssima e o restante como regular.

Para a estimativa da nota de facilidade e nota de documentação foi realizada a correlação do nível de experiência do participante com a nota obtida. Para esta estimativa, foi realizado a média com o cálculo de desvio de padrão, que pode ser verificado na tabela 5. Para o cálculo foi dado os valores 0 para documentação péssima, 1 para regular e 2 para ótima.

Nível de Experiência		Facilidade	Nota da documentação
Básico	Nota	6,75	Péssima
	Desvio Padrão	2,872	0
Intermediário	Nota	7,20	Péssima~Regular
	Desvio Padrão	1,483	0,577
Avançado	Nota	8,00	Não utilizaram
	Desvio Padrão	0	Não utilizaram
Total	Nota	7,18	Péssima~Regular
	Desvio Padrão	1,888	0,500

Tabela 5 - Média com desvio padrão

Sendo assim, percebe-se que o nível de experiência influencia na facilidade de utilização da biblioteca, porém verificando pela distância entre as notas não há muita

abrangência entre os casos. Pelo fato que os participantes com nível de experiência básico e intermediário recorrerem mais a documentação, e a mesma ser classificada como péssima pela maioria, entende-se que se a mesma fosse mais completa e intuitiva, estes participantes poderiam obter maior facilidade e alcançar os participantes mais avançados.

4.2.2 Avaliação da interface natural da Trilha

O questionário para avaliação dos comandos naturais e a implementação da trilha foi realizada com integrantes do Laboratório de Educação Ambiental da UNIVALI.

Para responder este questionário, foi solicitado para que os participantes iniciassem a aplicação do projeto da Trilha do Morro da Cruz, iniciasse o reconhecimento de comandos naturais, e interagisse com um dos conjuntos de comandos.

O questionário solicitou que os participantes respondessem o nível de conhecimento com jogos ou com o dispositivo Kinect, e uma nota de zero a dez para: aprendizagem dos comandos naturais; facilidade e conforto na execução nos comandos naturais, e fluência obtida na execução. Ao final, o participante também deverá responder se encontrou alguma restrição ou dificuldade, informando qual, e um campo opcional para observações e sugestões.

É apresentada uma discussão comparando os resultados com os participantes que utilizaram o conjunto padrão e o conjunto alternativo.

Foi obtido retorno de quatorze participantes, sendo sete utilizando o conjunto padrão de comandos e sete o conjunto alternativo.

Os resultados estão divididos pelo grupo de comandos naturais testados. A tabela 6 demostra os resultados encontrados com o primeiro conjunto de comandos, separados por nível de conhecimento em jogos, ou dispositivo Kinect. A coluna dificuldade demonstra dentro da população, quantos encontram algum tipo de dificuldade. A partir destes dados é possível perceber que o nível de conhecimento em computação, jogos ou no dispositivo Kinect, influencia tanto na facilidade, como na aprendizagem dos comandos naturais.

Conhecimento		Aprendizagem	Facilidade	Fluência	Dificuldade
Básico	Nota	5,2500	5,7500	5,2500	Não se aplica
	Qtd	4	4	4	4
	Desvio Padr.	2,06155	1,89297	1,70783	0
Intermed	Nota	5,5000	7,0000	6,5000	Não se aplica
	Qtd	2	2	2	2
	Desvio Padr.	3,53553	0,00000	0,70711	0
Avançado	Nota	7,0000	7,0000	7,0000	Não se aplica
	Qtd	1	1	1	1
	Desvio Padr.	0	0	0	0
Total	Nota	5,5714	6,2857	5,8571	Não se aplica
	Qtd	7	7	7	7
	Desvio Padr.	2,14920	1,49603	1,46385	0

Tabela 6 - 1º conjunto de comandos naturais

Seguindo o mesmo padrão da tabela 6, a tabela 7 demonstra os resultados obtidos com o conjunto alternativo de comandos naturais. De acordo com as notas de facilidade e fluência e as observações dos participantes deste conjunto, a redução da facilidade e conforto no uso dos comandos naturais, deve-se a locomoção do personagem na trilha se tornar cansativa, pois se utiliza dos braços em uma posição que alongo do tempo torna-se exaustiva.

Conhecimento		Aprendizagem	Facilidade	Fluência	Dificuldade
Básico	Nota	5,6667	5,0000	4,3333	
	Qtd	3	3	3	3
	Desvio Padr.	0,57735	1,00000	0,57735	
Intermed	Nota	6,0000	5,0000	5,5000	
	Qtd	2	2	2	2
	Desvio Padr.	1,41421	0,00000	0,70711	
Avançado	Nota	5,0000	6,5000	6,5000	
	Qtd	2	2	2	2
	Desvio Padr.	0,00000	0,70711	0,70711	
Total	Nota	5,5714	5,4286	5,2857	
	Qtd	7	7	7	7
	Desvio Padr.	0,78680	0,97590	1,11270	

Tabela 7 - 2º conjunto de comandos naturais

5 CONCLUSÃO

A partir dos dados obtidos na seção 4 sobre a usabilidade da interface, é possível perceber que o nível de conhecimento em computação, jogos ou no dispositivo Kinect, influencia tanto na facilidade do uso, como na aprendizagem dos comandos naturais. Os usuários com mais conhecimento nas áreas citadas acima se adaptam mais rapidamente por conhecerem as limitações do dispositivo, seja por questões de espaço ou da necessidade de precisão do comando natural a ser realizado. O que não seria o ideal para uma interface natural, na qual esta deveria se adaptar ao usuário e não o usuário a ela.

A partir de observações de alguns participantes durante os testes, percebeu-se grande dificuldade na memorização dos comandos naturais e na realização destes comandos. Pelo fato da interface natural ter sido desenvolvida para suprir todos os comandos não naturais, uma grande diversidade de poses teve de ser criada, dificultando para a memorização do usuário. Além disto, surgiu a necessidade de restringir a margem de erro do reconhecimento das poses, para que as poses não conflitassem entre si, o que tornou dificultosa a execução dos comandos por usuário em um primeiro momento.

Outro empecilho para o reconhecimento é a falta de *feedback* da imagem do usuário no momento da interação com a trilha. O usuário não consegue saber se seu corpo está dentro da área de reconhecimento do Kinect, e nem comparar o esqueleto sendo reconhecido com a pose a ser efetuada. O *feedback* se torna indispensável, inclusive para os comandos de voz para que o usuário entenda que comando recém falado não foi reconhecido.

Pelas notas do questionário e por observações notou-se que participantes que utilizaram conjunto alternativo de comandos, tiveram mais dificuldade e desconforto na utilização da interface natural. Isso ocorre pelo fato da locomoção do personagem na trilha se tornar cansativa com a utilização dos braços em uma posição exaustiva.

Após a realização da avaliação da trilha com usuários, foi possível perceber que o cenário ideal para uma trilha virtual, seria a utilização da quantidade de

comandos naturais mais reduzida. Desta forma poderia ser aumentada a margem de erro utilizada no reconhecimento da pose para facilitar o reconhecimento e dispensar a necessidade de um tutorial para memorização. Caso o tutorial seja realmente necessário, seria também ideal modificar o feedback atual, que mostra a exibição do esqueleto realizando tal pose, para a imagem real de uma pessoa realizando a mesma pose, permitindo assim que a noção de profundidade da pose possa ser percebida. Atualmente está fica prejudicada por a pose é representada somente através das linhas que representam o esqueleto.

Outros tipos de tutoriais mais interativos também são possíveis, como por exemplo, a cada nova interação, uma pequena janela se abre informando como o usuário pode proceder, demonstrando a pose, comando de voz ou gesto a ser realizado. Também seria possível adicionar esta funcionalidade juntamente com os feedbacks da interação via mouse e teclado, como por exemplo, inserir juntamente com a mensagem na qual informa que para fechar uma interação visual deve-se pressionar a tecla *enter*, o comando natural que pode ser realizado.

Algumas restrições do Kinect vieram a ser relatadas pelos usuários da trilha, dentre delas o reconhecimento de voz somente na língua inglesa e a dificuldade do reconhecimento dos comandos devido a pronúncia, seja pela presença ruídos no ambiente ou pela pronuncia incorreta da palavra. A limitação da língua para reconhecimento de voz é um grande empecilho para trabalhos educativos; muitos possuem dificuldade em pronunciar corretamente na língua inglesa, tornando o reconhecimento falho.

Outro fator limitante em relação ao dispositivo Kinect diz respeito a necessidade de espaço e luminosidade para a utilização do mesmo, o que torna difícil a utilização em ambientes públicos e mal iluminados.

Comparando os resultados de facilidade da biblioteca KinectProject obtidos entre os níveis de conhecimento em programação, percebe-se que não há muita dificuldade para a implementação das funcionalidades com um conhecimento básico em programação. Com apenas 3 comandos é possível iniciar um evento de reconhecimento, obter os eventos da fila e verificar se determinado comando foi reconhecido. Outra informação importante é que 63% dos participantes dos

questionários não necessitaram recorrer à documentação, evidenciando uma boa arquitetura e estrutura da biblioteca.

O que também dificulta a aprendizagem na utilização da biblioteca é a falta de uma documentação mais completa, com exemplos e informações. A documentação é indispensável, para que os desenvolvedores tenham ao que recorrer quando necessitarem de apoio.

Uma contextualização mais profunda antes da realização do questionário com informações sobre a biblioteca, sua estrutura, qual a sua finalidade e o sobre o Unity, ambiente no qual foram realizados os testes, poderia facilitar o entendimento dos participantes e assim consequentemente aumentar as notas de facilidade.

É necessário corrigir as falhas descritas na seção 4 a fim de facilitar e tornar possível adaptar diversos tipos de aplicações para uma interface natural utilizando o Kinect. A correção da funcionalidade do esqueleto do usuário ou exibição da imagem RGB é primordial para *feedbacks*. A verificação e possível correção do reconhecimento de poses também é importante, para melhorar a performance, pois aplicações como jogos muitas vezes necessitam alto desempenho.

A biblioteca KinectProject possui várias funcionalidades úteis para a implementação de uma interface natural, como a possibilidade de pausar e retomar os eventos de reconhecimento para evitar processamento adicional indesejável. Entretanto outras se tornam necessárias para facilitar a adaptação das interfaces naturais a diversos tipos de aplicações, como discutido nos próximos parágrafos.

O fato do KinectServer ser uma aplicação e não um serviço, torna inviável para algumas aplicações utilizar esta biblioteca. Seria necessário modificá-lo como um serviço para que aplicações pudessem ser distribuídas normalmente, sem a necessidade da utilização de um segundo software.

Outra função interessante seria a adequação automática do ângulo do Kinect, baseando-se no esqueleto do usuário em reconhecimento. Essa função dispensaria que o usuário necessitasse configurar o ângulo manualmente.

Apesar de correções da biblioteca não fazerem parte do objetivo e escopo deste trabalho, grande parte do esforço realizado foi para correções das mesmas. As

correções foram realizadas à medida que se tornaram necessárias e que não prejudicassem o cronograma deste trabalho, a fim de tornar possível a implementação da interface natural projetada. Trechos de código das alterações realizadas podem ser visualizados no apêndice J.

Com as correções e melhorias apresentadas na seção 4 devidamente realizadas, a biblioteca KinectProject tem grande potencial, sendo possível implementar novas funcionalidades facilmente, e assim se transformar em uma boa ferramenta de integração com o Kinect.

5.1 OBJETIVOS ALCANÇADOS

Nesta seção estão listados os objetivos específicos deste TTC e como os mesmos foram alcançados.

- Estudar e documentar a biblioteca KinectProject desenvolvida pelo Graduado em Computação Gabriel Schade:
  - ✓ Foi realizado o estudo do TTC realizado pelo Gabriel, da implementação do KinectProject, e documentado na seção 2.4
- Estudar e documentar a *engine* do Unity 3D:
  - ✓ Foram realizados estudos na no conceito e API do Unity, realizados testes e a documentação destes, conforme a seção 2.3
- Estudar e documentar o Projeto Trilha do Morro da Cruz feita no Unity 3D:
  - ✓ Estudos na implementação do Projeto da Trilha foram documentados na seção 2.5
- Estudar e documentar conceitos de ergonomia, usabilidade e acessibilidade relacionados a NUIs:
  - ✓ Objetivo cumprido, os estudos realizados foram documentados na seção 2.6

- Selecionar poses e comandos de voz adequados à interação com o ambiente virtual, baseando-se nos estudos e análises descritos nas etapas anteriores:
    - ✓ As poses selecionadas para a utilização na trilha virtual podem ser visualizadas no apêndice H;
- Projetar a interface natural proposta:
    - ✓ O projeto foi realizado da interface natural foi realizada.
- Projetar análises e testes a serem efetuados para validação da ferramenta:
    - ✓ Projeto e realizado as validações.
- Produzir a documentação do projeto da interface:
    - ✓ A documentação do projeto pode ser verificada na seção 3 de desenvolvimento.
- Implementar a interface natural, através do KinectProject, no projeto da Trilha do Morro da Cruz:
    - ✓ O projeto da interface natural foi implementada com sucesso e está documentada na seção 3.1
- Implementar um tutorial interativo:
    - ✓ Tutorial desenvolvido conforme detalhes na seção 3.1.2
- Executar testes e análises de verificação do funcionamento da interface e ferramenta:
    - ✓ Testes foram realizados, e os resultados podem ser verificados na seção 4
- Produzir um manual de instalação para usuários da nova interface natural do projeto da Trilha Virtual:
    - ✓ O manual foi realizado e pode ser visualizado no apêndice D.

- Validar a usabilidade da interface natural através da implantação do projeto no Laboratório de Educação Ambiental – LEA da UNIVALI:

    - ✓ Realizada a validação da interface com integrantes do laboratório, tornando o objetivo concluído;

- Documentar todas as etapas envolvidas no desenvolvimento deste TTC na forma de um relatório final e possível artigo científico:

    - ✓ Objetivo concluído com a entrega deste relatório final e do artigo científico;

Sendo assim, o objetivo geral deste trabalho que é a validação da biblioteca KinectProject, foi concluída.

5.2 TRABALHOS FUTUROS

Algumas necessidades e sugestões estão comentadas nesta seção, a fim de corrigir e melhorar a biblioteca KinectProject.

1. Inserir identificador em cada pose quando esta é armazenada através do Editor de Poses - InputRegisterApp, para que se possa passar para o KinectServer, quais poses querem ser testadas. Da forma atual, todas as poses da pasta são testadas, sendo que uma aplicação pode não estar ou não querer utilizar todas em um determinado momento, sendo que a única forma possível de contornar é a distribuição dos comandos em pastas diferentes, parar os eventos de reconhecimento e iniciar novamente;

2. Tratar as exceções do InputRegisterApp, com mensagens de aviso em vez do travamento da aplicação, o mesmo vale para o KinectServer;

3. Criar uma fila para alteração do ângulo do Kinect para resolver a exceção levantada caso uma nova solicitação de alteração seja recebida antes do fim da primeira;

4. Criar uma fila separada para os eventos do esqueleto do usuário, para que os eventos de pose, voz e gesto sejam reconhecidos simultaneamente. Ou então mudar da estrutura de fila para uma outra estrutura, para que assim dois eventos possam ser reconhecidos simultaneamente, sem necessidade de remover o item da fila para depois reconhecer o próximo;

5. Corrigir a transmissão de dados da câmera RGB para as aplicações via KinectServer, para que a imagem possa ser exibida;

6. Criar reconhecimento de gestos personalizados, com a possibilidade de cria-los através do InputRegisterApp;

7. Verificar o motivo do cálculo para reconhecimento de poses via produto escalar não estar funcionando corretamente comparado com a execução pela lei dos cossenos, a fim de melhorar o desempenho da aplicação;

8. Verificar e analisar uma forma de diminuir o consumo de hardware da aplicação KinectServer;

9. Analisar e modificar se possível, o aplicativo KinectServer para um serviço do sistema operacional, sem a necessidade da intervenção do usuário final;

10. Adicionar funcionalidade de alteração automática do ângulo de visão do Kinect para a melhor posição baseando-se no esqueleto do usuário.

11. Permitir a possibilidade do InputRegisterApp ser aberto enquanto o KinectServer estiver em execução;

12. Complementar a documentação com exemplos de cada comando e como usar. Escrever em diferentes línguas não somente inglês; e

13. Modificar o tutorial para se tornar mais intuitivo conforme as conclusões descritas neste projeto.

5.3 CONCLUSÕES FINAIS

Com este trabalho foi possível entender a melhor forma de desenvolver uma interface natural. Nem sempre é aconselhável transformar todos os comandos existentes em uma interface gráfica em comandos naturais. A grande diversidade de comandos dificulta a aprendizagem do usuário e também torna exaustiva a interação.

Como as aplicações variam em seus requisitos de interação, estas podem utilizar-se de diferentes dispositivos que permitem a implementação de uma interface natural. Para o dispositivo Kinect, podem-se ter jogos com o objetivo de se realizar um grande esforço físico como é o caso de alguns jogos de esportes, e também ambientes virtuais, desde que suas interações não sejam exaustivas.

Com a constante evolução dos dispositivos de interação não convencionais como o Kinect que, por oferecer um SDK aberto, já está sendo utilizado inclusive em aplicações web, torna-se interessante para desenvolvedores tomarem conhecimento do funcionamento das interfaces naturais, de ferramentas que auxiliam nesta área, e começarem a se adequar aos novos padrões de usabilidade e implementações em interface natural.

Baseando-se nas informações obtidas e descritas neste projeto, juntamente com o fato de não existir outra ferramenta disponível com as mesmas funcionalidades, a biblioteca KinectProject torna-se de grande utilidade e com potencial para auxiliar desenvolvedores a migrar suas aplicações para uma interface natural.

REFERÊNCIAS

BALSAMIQ. **Balsamiq Mockups**. 2013. Disponível em: <http://www.balsamiq.com/>. Acesso em: 17 mar. 2013.

CYBORG, Anthropology. **Steve Mann**. 2013. Disponível em: <http://cyborganthropology.com/Steve_Mann>. Acesso em: 05 jun. 2013.

DAM, Peter; LOAIZA, Manuel; SOARES, Luciano; RAPOSO, Alberto. **A Study of Selection and Navigation Techniques Using Kinect in VR**. 2012. Disponível em: <http://www.tecgraf.puc-rio.br/~abraposo/pubs/SVR2012/Short_Peter.pdf>. Acesso em: 15 maio 2013.

GAGE. **Getting to Know NUI**. 2013. Disponível em: <http://www.gage.com/connect/insights/2013/gettingtoknownui.aspx/>. Acesso em: 02 dez. 2013.

GALEOTE, Sidney. **Tipos de teste de software**. 2011. Disponível em: <http://www.galeote.com.br/blog/2011/06/tipos-de-testes-de-software/>. Acesso em: 05 maio 2013.

IBM. **IBM SPSS software.** 2013. Disponível em: < http://www-01.ibm.com/software/analytics/spss/>. Acesso em: 20 set. 2013.

IEA, International Ergonomics Association. **What is Ergonomic?** 2010. Disponível em: <http://www.iea.cc/01_what/What%20is%20Ergonomics.html>. Acesso em: 14 mar. 2013.

JORDAN, Gustavo. **Software Livre do Brasil – Interface Natural do Usuário**. 2013. Disponível em <http://softwarelivre.org/jordan/blog/interface-natural-do-usuario-nui-%E2%80%93-natural-user-interface>. Acesso em: 20 abr. 2013.

MICHAELIS UOL. **Dicionário Online Michaelis**. 2013. Disponível em: <http://michaelis.uol.com.br/>. Acesso em: 09 mar. 2013.

MICROSOFT HIG. **Kinect for Windows Human Interface Guidelines**. 2013. Disponível em: < http://go.microsoft.com/fwlink/?LinkID=247735>. Acesso em 09 mar. 2013.

MICROSOFT KINECT. **Kinect Documentation& Resources**. 2013. Disponível em: <http://www.microsoft.com/en-us/kinectforwindows/develop/resources.aspx>. Acesso em 09 mar. 2013.

MICROSOFT MSDN. **Visual C#**. 2013. Disponível em: < http://msdn.microsoft.com/pt-BR/vstudio/hh388566>. Acesso em: 13 mar. 2013

MICROSOFT RESEARCH, Research. **NUI: Natural User Interface**. 2013. Disponível em: <http://research.microsoft.com/en-us/collaboration/focus/nui/default.aspx>. Acesso em 09 mar. 2013.

NINTENDO. **Wii Official Site at Nintendo**. 2013. Disponível em: <http://www.nintendo.com/wii>. Acesso em: 13 mar. 2013

NORMAN, Donald. **Gestural Interfaces: A Step Backwards in Usability**. 2012. Disponível em: <http://www.jnd.org/dn.mss/gestural_interfaces_a_step_backwards_in_usability_6.html>. Acesso em: 08 mar. 2013.

OMG. **Object Management Group – UML**. 2013. Disponível em: <http://www.uml.org>. Acesso em: 20 maio 2013.

RAYMOND, Eric. **The first GUIs.** 2013. Disponível em: <http://www.catb.org/~esr/writings/taouu/html/ch02s05.html>. Acesso em: 27 maio 2013.

REIS, Eberton. **TestExpert - Estratégia, Tipos e Técnicas de teste**. 2009. Disponível em: <http://www.testexpert.com.br/?q=node/1375>. Acesso em :16 maio 2013.

SANTOS, Robson L. G. **Usabilidade e métodos de avaliação de usabilidade de linterfaces web**. Rio de Janeiro. Pontifícia Universidade Católica do Rio de Janeiro. Disponível em: <http://pt.scribd.com/doc/50857920/Usabilidade-e-Metodos-de-Avaliacao-de-Usabilidade-de-Interfaces-WEVB//pt.scribd.com/doc/50857920/Usabilidade-e-Metodos-de-Avaliacao-de-Usabilidade-de-Interfaces-WEVB>. Acesso em: 20 maio 2013.

SCHADE, Gabriel UNIVERSIDADE DO VALE DO ITAJAÍ. **Biblioteca de Funções para utilização do Kinect em Aplicações GUI e Jogos Eletrônicos**. 2012. TCC (graduação em Ciência da Computação) - Universidade do Vale do Itajaí, Itajaí, 2012. Disponível em : <http://Siaibib01.univali.br/pdf/Gabriel Schade Cardoso.pdf>. Acesso em: 21 fev. 2013.

SILVA, Eduardo A. **Trilha virtual interativa em 3D do Morro da Cruz: Disseminação digital do conhecimento obtido no processo de restauração da biodiversidade do local**. Itajaí, 2012. Universidade do Vale do Itajaí, CTTMAR – Centro de Ciências Tecnológicas da Terra e do Mar, Brasil. Disponível em: <http://sbgames.org/sbgames2012/proceedings/papers/cultura/C_S10.pdf>. Acesso em: 08 mar. 2013.

SONY. **Playstation Move Motion**. 2013. Disponível em: <http://us.playstation.com/ps3/playstation-move/>. Acesso em: 13 mar. 2013.

SPARX System. **UML Tools for Software Development**. Disponível em: <http://www.sparxsystems.com.au/>. Acesso em: 04 jun. 2013.

SZNELWAR Laerte; ABRAHÃO, Júlia; SILVINO, Alexandre; SARMET, Maurício; PINHO, Diana. **Introdução à Ergonomia: da Prática à Teoria**. Editora Blucher, 2009.

TURK, TRT. ***İşte dokunmatik çeşitleri.*** 2013. Disponível em: <http://www.trtturk.com.tr/haber/iste-dokunmatik-cesitleri.html>. Acesso em: 10 abr. 2013.

UNITY Technologies. **Unity Documentation**. 2013. Disponível em: <http://unity3d.com/company/support/documentation/>. Acesso em: 09 mar. 2013.

UNIVALI, Morro da Cruz. **Morro da Cruz Univali.** 2013. Disponível em: <http://morrodacruzunivali.blogspot.com.br/>. Acesso em: 20 mar. 2013.

ZIGFU. **Unity Recipes**. 2013. Disponível em: <http://zigfu.com/en/zdk/recipes/>. Acesso em: 26 maio 2013.

APÊNDICE A. CENÁRIOS E CASOS DE USO

Neste apêndice estão os cenários dos casos de uso referentes a implementação do Projeto da Trilha do Morro da Cruz gerados pela ferramenta Enterprise Architect.

A.1 CASOS DE USO

Os casos de usos estão divididos em: funcionalidades acessíveis pelo ambiente e acessíveis pelo menu.

Os casos de uso do menu podem ser vistos na figura 27, dentro dele estão: iniciar ou continuar passeio; alterar as configurações de entrada; testar e aprender as entradas configuradas; e reiniciar passeio.

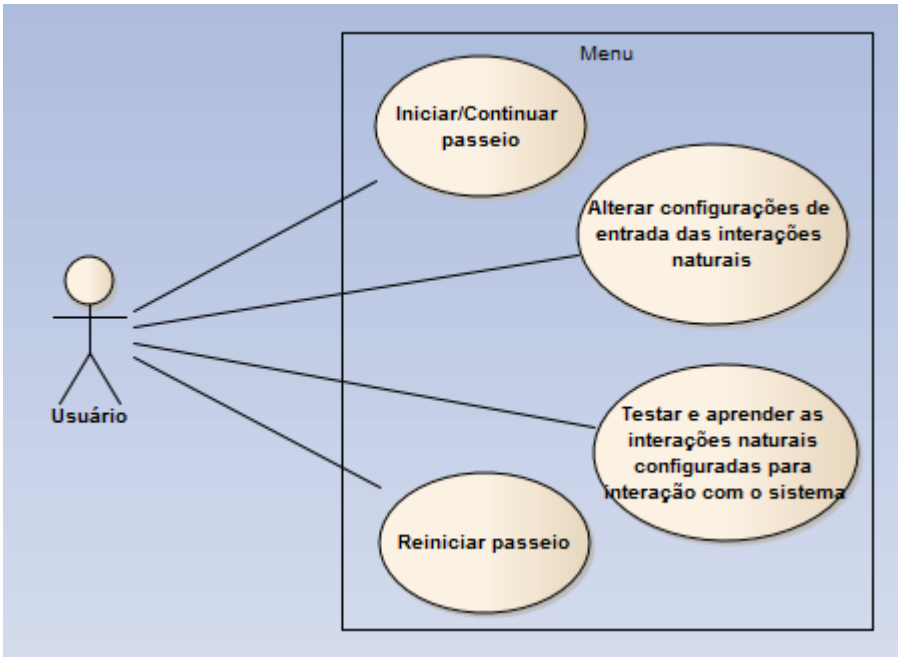

Menu
Iniciar/Continuar passeio
Alterar configurações de entrada das interações naturais
Testar e aprender as interações naturais configuradas para interação com o sistema
Reiniciar passeio
Usuário

Figura 26. Casos de uso menu

Já os casos de uso do ambiente, visualizados na figura 28, são: acessar menu ou pausar; interagir com objetos visuais; interagir com objetos textuais; e locomover-se no ambiente.

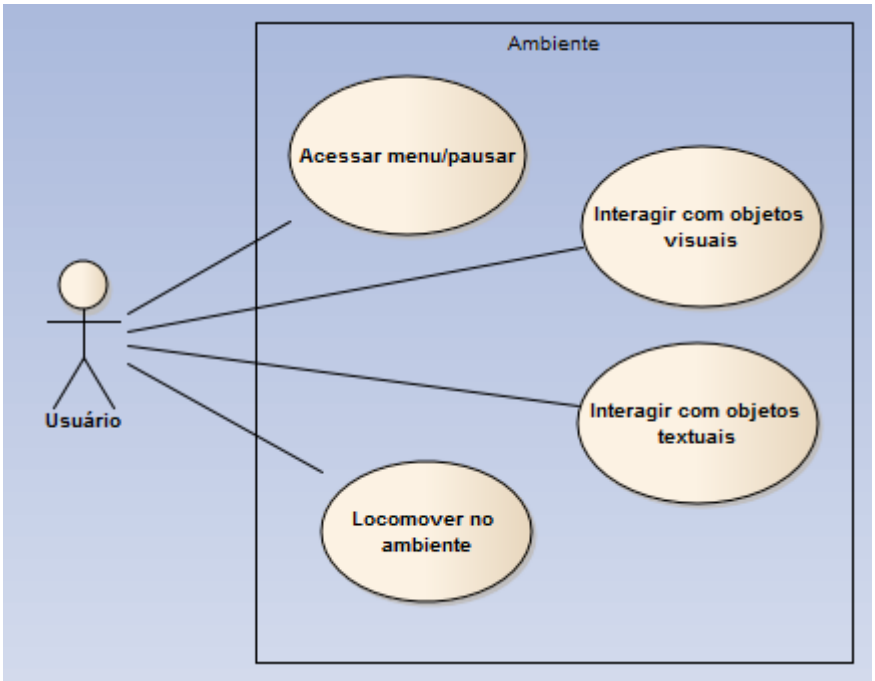


Figura 27. Casos de uso ambiente

A.2 CENÁRIOS

Iniciar/Continuar passeio

Cenário Principal

1. O usuário executa a interação natural que ativa o início ou continuação do passeio;
2. O sistema reconhece o comando;
3. O sistema fecha o menu principal;
4. O sistema carrega as interações naturais do ambiente de passeio.

Reiniciar passeio

Cenário Principal

1. O usuário executa a interação natural que reinicia o passeio;
2. O sistema reconhece o comando;
3. O sistema apresenta a mensagem "Passeio reiniciado";
4. O sistema recarrega a posição inicial do passeio;
5. O sistema retorna ao menu principal.

Acessar menu/pausar

Cenário Principal

1. O usuário executa a interação natural que ativa o menu/pausa;
2. O sistema reconhece o comando;
3. O sistema abre o menu principal;
4. O sistema carrega as interações naturais do menu.

Interagir com objetos textuais

Cenário Principal

1. O usuário executa o comando para interagir com objetos no ambiente de passeio;
2. O sistema reconhece o comando e o objeto a ser interagido;
3. O sistema apresenta as informações textuais;
4. O sistema carrega as entradas na interação com objetos;
5. O sistema aguarda o usuário executar o comando natural de sair da interação;
6. O usuário executa o comando de interação como swipe para rolar o texto e retorna ao passo 5;
7. O sistema retorna ao ambiente de passeio;
8. O sistema recarrega as entradas de interação com o ambiente de passeio.

Sair da Interação

Caso o comando para sair da interação seja utilizado, ir para o passo 7.

Interagir com objetos visuais

Cenário Principal

1. O usuário executa o comando para interagir com objetos no ambiente de passeio;
2. O sistema reconhece o comando e o objeto a ser interagido;
3. O sistema apresenta a imagem no ambiente;
4. O sistema carrega as entradas na interação com objetos;
5. O sistema aguarda o usuário executar o comando natural de sair da interação;
6. O usuário executa o comando de interação;
7. O sistema retorna ao ambiente de passeio;
8. O sistema recarrega as entradas de interação com o ambiente de passeio.

APÊNDICE B. LOG DO KINECT SERVER

O KinectServer contém em sua tela uma área reservada para o log de eventos despachados. Todos os eventos despachados pelo cliente do KinectServer são inseridos em uma fila e executados em sequência, ambos são registrados no log.

Cada registro possui a hora, minuto, segundo e milissegundos no qual foi executado ou recebido pelo servidor. Não é necessário acionar o log, o mesmo é executado automaticamente e sem a possibilidade de desabilita-lo. Na figura 26, é demonstrado o log no qual uma aplicação cliente do KinectServer envia uma requisição para iniciar a captura de gestos. Logo depois solicita que o servidor tente obter novos eventos e verifica que não há novos eventos para serem enviados para a aplicação cliente.

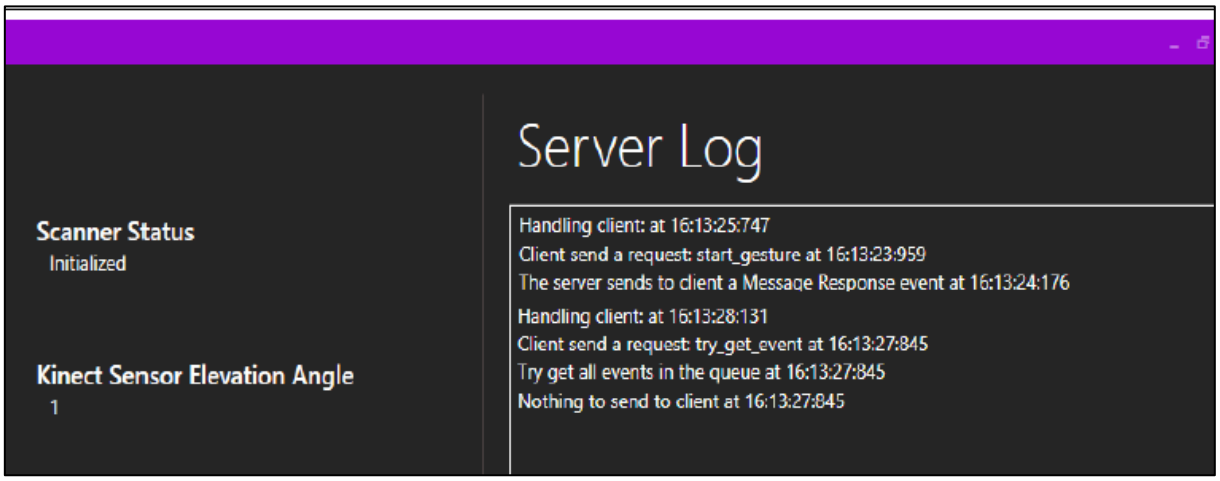

Server Log
Scanner Status
Initialized
Kinect Sensor Elevation Angle
1
Handling client: at 16:13:25:747
Client send a request: start_gesture at 16:13:23:959
The server sends to client a Message Response event at 16:13:24:176
Handling client: at 16:13:28:131
Client send a request: try_get_event at 16:13:27:845
Try get all events in the queue at 16:13:27:845
Nothing to send to client at 16:13:27:845

Figura 28. Log do KinectServer
Fonte: Schade, Gabriel (2013).

APÊNDICE C. LOG DO UNITY

O Unity conta com um *console* para uso dos desenvolvedores para fazer o *debug* das cenas. Os desenvolvedores podem utiliza-lo para demonstrar mensagens em algumas áreas de código, assim como retornar valores de objetos ou variáveis.

Através da classe Debug, utiliza-se o método Log para exibir as mensagens. A figura 27 demonstra o uso deste método para demonstrar a frase “Pulando” no *console* do Unity. O código foi inserido no método do controle de um componente no qual identifica quando é pressionado a tecla de espaço para pular.

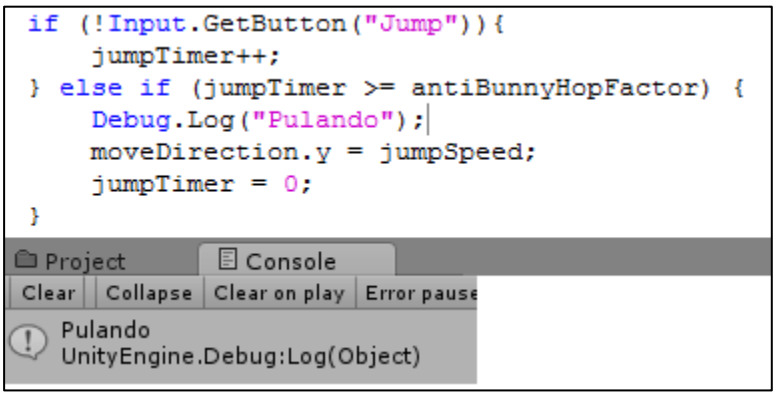


Figura 29. Debug no console do Unity

APÊNDICE D. MANUAL DE INSTALAÇÃO E CONFIGURAÇÃO DA TRILHA

1. Instalação

Execute o arquivo de instalação do Projeto da Trilha, o mesmo será descompactado na pasta Arquivos de Programas da unidade do Sistema Local por padrão. O projeto contém o aplicativo do ambiente da trilha desenvolvido no Unity, o aplicativo KinectServer responsável pela comunicação com o Kinect, e também o aplicativo InputRegisterApp, para criação/edição de poses e comandos de voz.

2. Configurando

Todos os comandos de interação natural são salvos na pasta de instalação do projeto. Os comandos de voz se encontram na pasta VoiceCommands e as poses na pasta Poses. Utilizando o aplicativo InputRegisterApp pode-se criar novas poses ou comandos de voz, e deve-se salvar os mesmos em suas respectivas pastas do projeto.

3. Criação das poses e comandos de voz

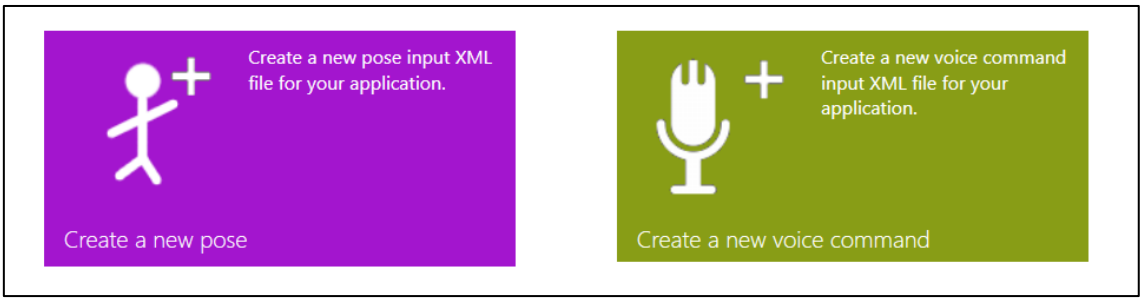

Create a new pose input XML file for your application.
Create a new pose
Create a new voice command input XML file for your application.
Create a new voice command

Figura 30. Menu InputRegisterApp

Abra o aplicativo InputRegisterApp, vá em "*Create a new pose*". Escreva o nome da pose e posicione-se para gravação. Ao estar posicionado diga as palavras Kinect e logo após Record Pose.

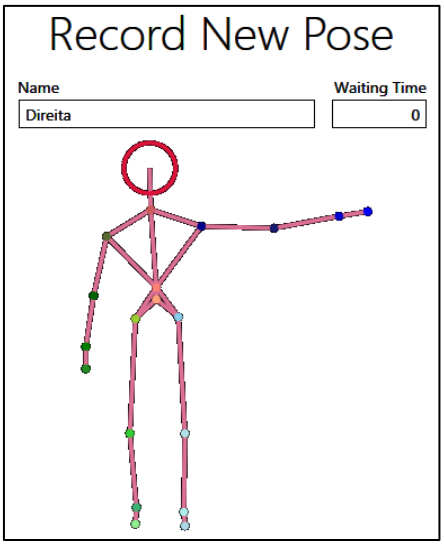


Figura 31. Pose reconhecida no InputRegisterApp

Grave pelo menos uma subpose, e salva a pose criada. Você pode ir em *Settings* para mudar o local onde a pose será salva.

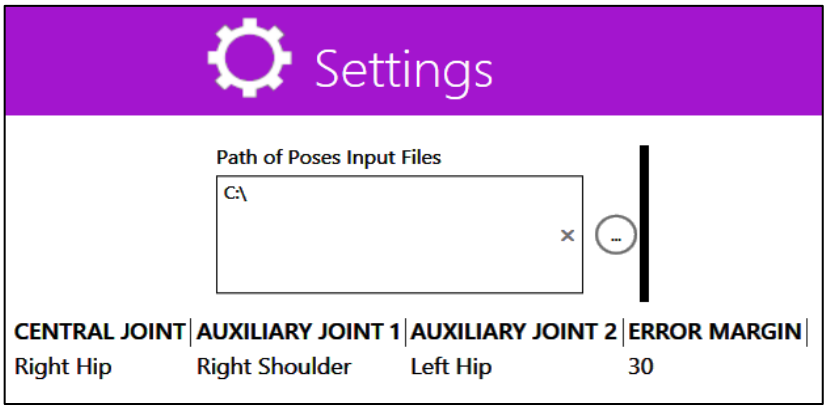


Figura 32. Configurações do InputRegisterApp

Para os comandos de voz vá em "*Create a new voice command*". Crie um nome para o grupo de comandos de voz, insira dois comandos de voz em inglês para reconhecimento.

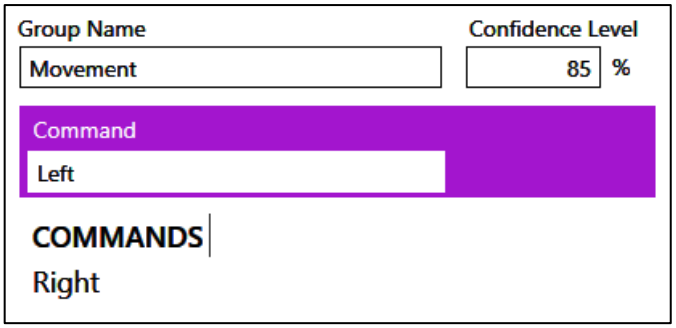


Figura 33. Criado um novo comando de voz

É possível para cada comando de interação no projeto, selecionar um comando de voz, uma pose, e um gesto. A alteração desses comandos é possível pelo menu “Config. Kinect” acessível pela aplicação da trilha. Através de uma caixa de seleção, é possível informar se o tipo de reconhecimento está ativo ou não. Já no campo de texto, deve-se informar o nome do comando cadastrado na aplicação InputRegisterApp. No caso de uma pose utilize o nome cadastrado no campo Name na hora do registro, no comando de voz, a palavra ou frase em si e para os gestos escreva “Right Swype” ou “Left Swype”.

4. Execução

Para a execução do projeto da Trilha do Morro da Cruz, é sempre necessário primeiramente:

- Conectar o Microsoft Kinect ao computador;
- Executar o aplicativo KinectSever;

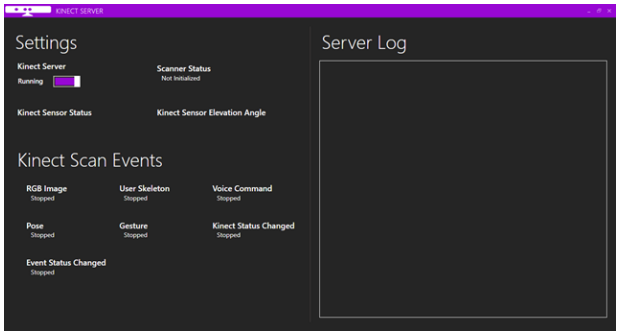


Figura 34. KinectServer

- Iniciar o reconhecimento no KinectServer; e
- Executar o Projeto da Trilha do Morro da Cruz

APÊNDICE E. QUESTIONÁRIO DE VALIDAÇÃO DE USABILIDADE

Questionário de Avaliação do Projeto Trilha do Morro da Cruz com uso da interface natural

Comandos de interação:
( ) Padrão ( ) Alternativo

Como você considera seu nível de conhecimento em computação, jogos ou no uso do dispositivo Kinect?

( ) Básico ( ) Intermediário ( ) Avançado

Dê uma nota de 0 a 10 para as próximas considerações:

Aprendizagem dos comandos naturais: _____

Facilidade e conforto no uso dos comandos naturais: _____

Fluência no uso da aplicação: _____

Encontrou dificuldade na utilização da trilha ou alguma restrição?

( ) Sim ( ) Não

Se sim, comente:

_____
_____
_____
_____

Observações e sugestões:

_____
_____
_____
_____
_____
_____

Quadro 5. Questionário de usabilidade

APÊNDICE F. VALIDAÇÃO DA BIBLIOTECA

Este documento trata-se de um passo a passo da utilização da biblioteca KinectProject, que integra o Microsoft Kinect com diversas aplicações, para a avaliação da mesma. Para avaliação, iremos desenvolver uma interface natural para um mini jogo no Unity, pois a biblioteca já conta com a DLL de integração com o mesmo.

Os passos a serem executados neste documento são:

- Criação de duas Poses
- Criação de dois Comandos de Voz
- Obter os Eventos do KinectProject no Unity
- Identificar Poses no Unity
- Identificar Comandos de Voz no Unity
- Identificar Gestos no Unity

1. Mini Jogo: Jump Game

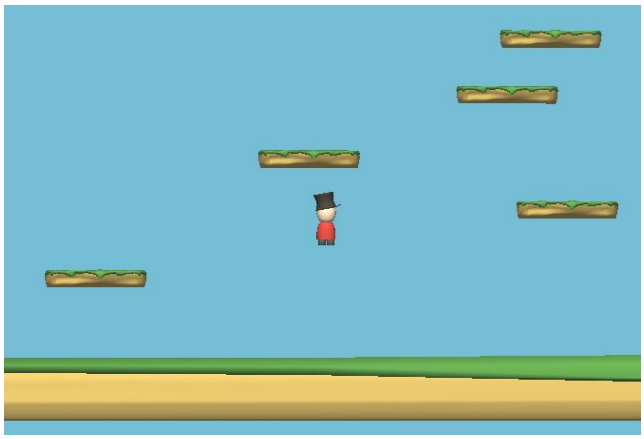


Figura 1 - Jump Game

Neste jogo, o personagem principal mantem-se pulando constantemente e tem como objetivo alcançar o topo do mapa subindo através de plataformas. O jogador deve movimentar o personagem para direita ou esquerda para completar o objetivo. Sendo assim, iremos criar interações para mover o personagem tanto para direita ou para esquerda.

2. Criação das poses e comandos de voz

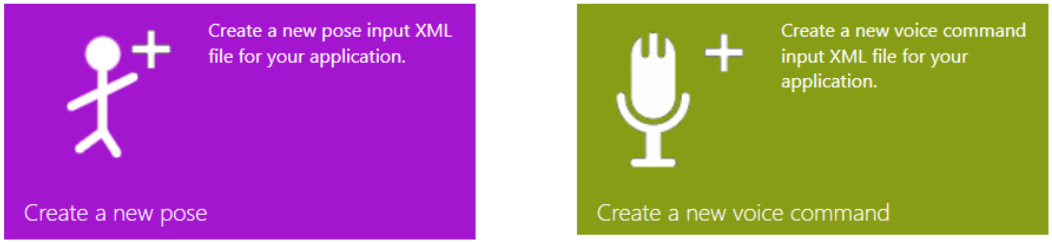

Create a new pose input XML file for your application.
Create a new pose
Create a new voice command input XML file for your application.
Create a new voice command

Figura 2- InputRegisterApp

Abra o aplicativo InputRegisterApp, vá em "*Create a new pose*". Escreva o nome da pose e posicione-se para gravação. Ao estar posicionado diga as palavras Kinect e logo após Record Pose.

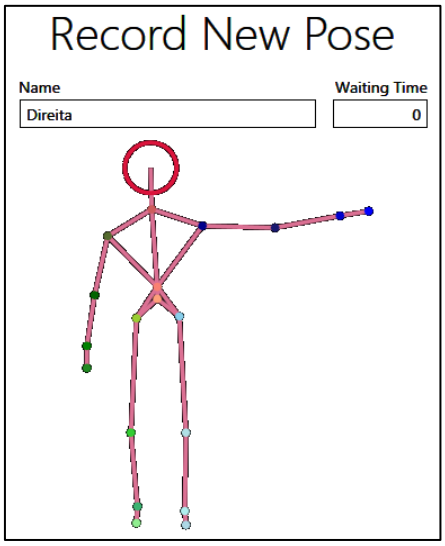


Figura 35. InputRegisterApp

Grave pelo menos uma subpose, e salva a pose criada. Você pode ir em *Settings* para mudar o local onde a pose será salva.

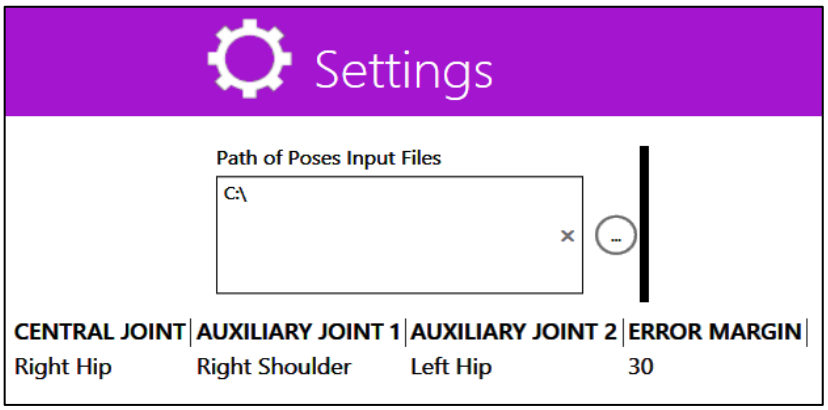


Figura 36. InputRegisterApp

Crie duas poses e comandos de voz distintos para cada movimento a ser realizado no jogo.

Agora vá em "*Create a new voice command*". Crie um nome para o grupo de comandos de voz, insira dois comandos de voz em inglês para reconhecimento no jogo.

Figura 37. InputRegisterApp

3. Configurando o Unity

Adicione as DLLs KinectUnity e KinectEvents na pasta *Plugins* do mini jogo. Salve as poses criadas em uma pasta chamada Poses e os comandos de voz em uma pasta chamada *VoiceCommands* dentro da pasta *Assets*.

Obter eventos gerados pelo Kinect/KinectProject

Abra o jogo no Unity e o arquivo *PlayerMovement.cs*, script no qual está adicionado ao personagem do jogo. Através deste arquivo faremos todo o controle do KinectProject e movimento do personagem.

Instancie o KinectInput no *Start* do *script* e envie requisições para iniciar os reconhecimentos como no exemplo do quadro 1:

```
01    KinectInput = KinectUnity.Classes.KinectInput.Instance;
02    KinectInput.SendRequest( new
KinectEvents.Classes.Communication_Classes.Requests.ClearEventQueueReq
uest());
03    KinectInput.SendRequest( new
KinectEvents.Classes.Communication_Classes.Requests.StartVoiceCommandR
equest(@"D:\Dados\Documentos\Unity\Assets" ) );
04    KinectInput.SendRequest( new
KinectEvents.Classes.Communication_Classes.Requests.StartPoseRequest(@
"D:\Dados\Documentos\Unity\Assets" ) );
05    KinectInput.SendRequest( new
KinectEvents.Classes.Communication_Classes.Requests.StartGestureReques
t(KinectEvents.Enums.Gestures.Left_Swipe));
06    KinectInput.SendRequest( new
KinectEvents.Classes.Communication_Classes.Requests.StartGestureReques
t(KinectEvents.Enums.Gestures.Right_Swipe));
```

Quadro 6. Iniciar Reconhecimentos

Dentro do método *Update* do *Script* adicione o seguinte código abaixo para obter os eventos da fila.

```
01    if( countToCheckNewEvents > 0.5 )
02    {
03          KinectInput.SendRequest( new
KinectEvents.Classes.Communication_Classes.Requests.TryGetEventsReques
t());
04          countToCheckNewEvents = 0;
05    }
06    else
07    {
08          countToCheckNewEvents += Time.deltaTime;
09    }
```

Quadro 7. Obter eventos na fila

Conforme o exemplo no quadro 3, utilize a KinectInput para verificar se determinado evento foi reconhecido. Crie as variáveis com o mesmo nome do quadro 3, pois as mesmas serão utilizadas para realizar a movimentação do personagem no Unity.

```
01    bool? speechRightMove =
KinectInput.GetInputVoiceCommand("Right");
02    bool? speechLeftMove  = ...
03
04    bool? poseRightMove = KinectInput.GetInputPose("Direita");
05    bool? poseLeftMove  = ...
06
07    bool? gestureRightMove =
KinectInput.GetInputGesture(KinectEvents.Enums.Gestures.Right_Swipe)
08    bool? gestureLeftMove  = ...
```

Quadro 8. Reconhecer evento

4. Realizando o teste

Para realizar o teste do que foi implementado, abra o aplicativo KinectServer, inicie o mesmo, e inicie a cena no Unity. Qualquer dúvida a documentação pode ser consultada para maiores informações.

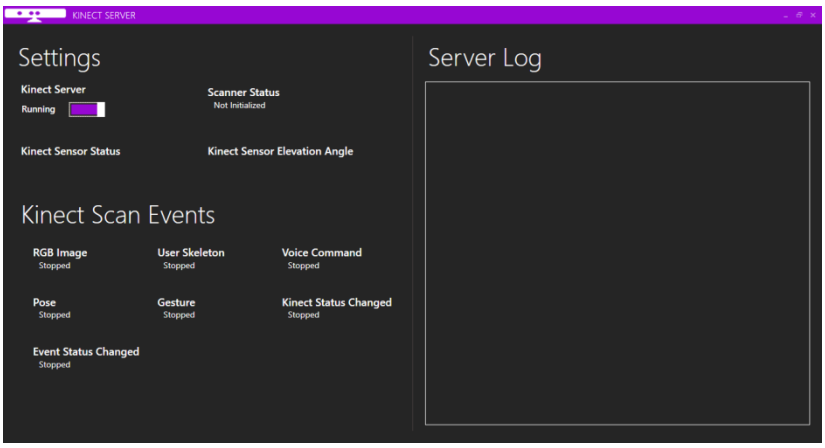


Figura 38. KinectServer

APÊNDICE G. QUESTIONÁRIO DE VALIDAÇÃO DA BIBLIOTECA

Questionário de Avaliação da Biblioteca Kinect Project

Nome (Opcional): _____

Como você considera seu nível de experiência e conhecimento em programação?

( ) Básico ( ) Intermediário ( ) Avançado

Conseguiu concluir o teste com sucesso?

( ) Sim ( ) Não

Recorreu a documentação da biblioteca?

( ) Sim ( ) Não

Se sim, como considera o nível da qualidade da documentação?

( ) Péssima ( ) Regular ( ) Ótima

Ocorreram erros?

( ) Sim ( ) Não

Se sim, quantos erros ocorreram: _____

Dê uma nota de 0 a 10 em relação a facilidade obtida para a implementação das funções: _____

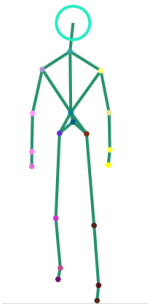
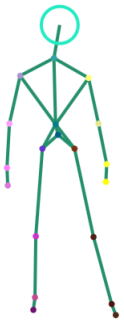
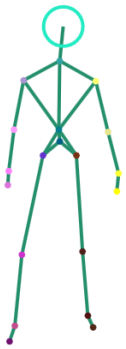
Observações ou sugestões:

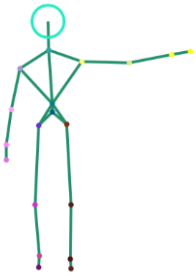
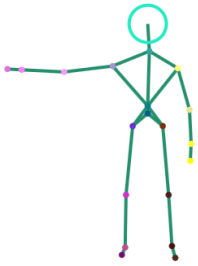
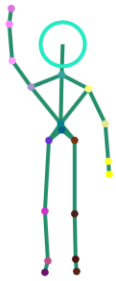
_____
_____
_____
_____
_____
_____

Quadro 9. Questionários de facilidade da biblioteca

APÊNDICE H. COMANDOS NATURAIS DEFINIDOS

A tabela 8 demonstra o conjunto padrão de todos os comandos mapeados para a interface natural.

Comando	Interação Atual	Interação Natural	Pose
Locomover à frente	Pressionar a tecla W	Pose: Pé direito à frente	
Locomover diagonal direita	Pressionar a tecla W e D juntas	Pose: Pé direito na diagonal direita ou ombros deslocados a direita	
Locomover diagonal esquerda	Pressionar a tecla W e A juntas	Pose: Pé esquerdo na diagonal esquerda ou ombros deslocados a esquerda	

Comando	Interação Atual	Interação Natural	Pose
Mover câmera à direita	Movimento do mouse para a direita	Gesto: Movimentar mão da direita para a esquerda Pose: Braço direito esticado à direita	
Mover câmera à esquerda	Movimento do mouse para a esquerda	Gesto: Movimentar mão da esquerda para a direita Pose: Braço esquerdo esticado à esquerda	
Acionar Interação (Por proximidade)	Não existente	Pose: Braço direito esticado para cima	
Sair da Interação	Pressionar tecla ESC	Pose: Posição que Simboliza o Pare com a mão direita ou esquerda Comando de voz: Out	
Trocar aba na Interação	Clique do mouse sobre a aba	Gesto: Movimentar mão da esquerda para a direita ou vice-versa	
Mover barra de rolagem lateral para leitura da interação	Clicar e arrasta o mouse sobre a barra de rolagem	Gesto: Movimentar mão de cima para baixo ou vice-versa	

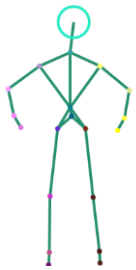
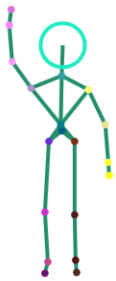
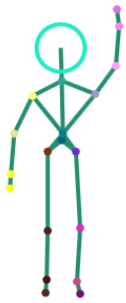
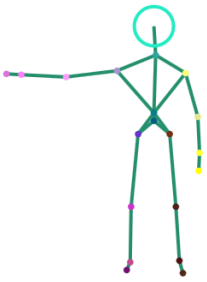
Comando	**Interação Atual**	**Interação Natural**	**Pose**
Pausar/Menu	Pressionar tecla ESC	Pose: Mãos na cintura Comando de voz: Menu	
Menu: Iniciar	Clique do mouse sobre o botão	Pose: Braço direito esticado para cima Comando de voz: Start	
Menu: Tutorial	Não existente	Pose: Braço esquerdo esticado para cima Comando de voz: Tutorial	
Menu: Reiniciar	Não existente	Pose: Posição que Simboliza o Pare com a mão direita ou esquerda Comando de voz: Restart	
Tela Menu: Configuração	Clique do mouse sobre o botão	Pose: Braço esquerdo esticado à esquerda	
Tela Tutorial: Próxima Interação	Não existente	Comando de Voz: Okay	

Tabela 8. Conjunto padrão de comandos naturais

A tabela 9 demonstra o conjunto alternativo de todos os comandos mapeados para a interface natural, as principais alterações foram na locomoção pela utilização do braço direito.

Comando	**Interação Atual**	**Interação Natural**
Locomover à frente	Pressionar a tecla W	Pose: Braço direito apontando para frente
Locomover diagonal direita	Pressionar a tecla W e D juntas	Pose: Braço direito apontando para a diagonal direita
Locomover diagonal esquerda	Pressionar a tecla W e A juntas	Pose: Braço direito apontando para a diagonal esquerda
Mover câmera à direita	Movimento do mouse para a direita	Gesto: Movimentar mão da direita para a esquerda Pose: Braço direito esticado à direita
Mover câmera à esquerda	Movimento do mouse para a esquerda	Gesto: Movimentar mão da esquerda para a direita Pose: Braço esquerdo esticado à esquerda
Acionar Interação (Por proximidade)	Não existente	Pose: Braço esquerdo esticado para cima
Sair da Interação	Pressionar tecla ESC	Pose: Posição que Simboliza o Pare com a mão direita ou esquerda Comando de voz: Out
Trocar aba na Interação	Clique do mouse sobre a aba	Gesto: Movimentar mão da esquerda para a direita ou vice-versa
Mover barra de rolagem lateral para leitura da interação	Clicar e arrasta o mouse sobre a barra de rolagem	Gesto: Movimentar mão de cima para baixo ou vice-versa

Comando	**Interação Atual**	**Interação Natural**
Pausar/Menu	Pressionar tecla ESC	Pose: Mãos na cintura Comando de voz: Menu
Menu: Iniciar	Clique do mouse sobre o botão	Pose: Braço esquerdo esticado para cima Comando de voz: Start
Menu: Tutorial	Não existente	Pose: Braço esquerdo esticado para cima Comando de voz: Tutorial
Menu: Reiniciar	Não existente	Pose: Posição que Simboliza o Pare com a mão direita ou esquerda Comando de voz: Restart
Tela Menu: Configuração	Clique do mouse sobre o botão	Pose: Braço esquerdo esticado à esquerda
Tela Tutorial: Próxima Interação	Não existente	Comando de Voz: Okay

Tabela 9. Conjunto alternativo de comandos naturais

APÊNDICE I. ESTRUTURA DE PASTAS DO PROJETO DA TRILHA

O Projeto da Trilha compilado e montado possui uma pasta chamada de data, onde todas as informações necessárias do projeto são armazenadas. A figura 40 demonstra a estrutura de pastas, onde o arquivo de comandos é armazenado na raiz. As poses devem ficar armazenadas na pasta Poses e os comandos de voz na pasta VoiceCommands.

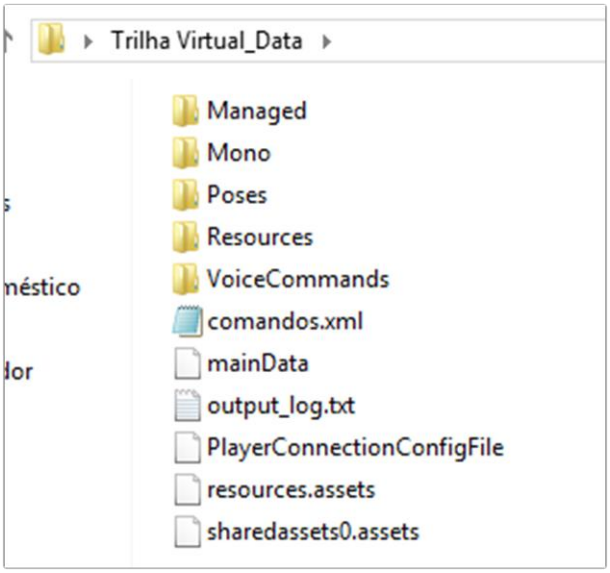


Figura 39. Estrutura de pastas do Projeto da Trilha

APÊNDICE J. TRECHOS DE CÓDIGO DAS CORREÇÕES E ALTERAÇÕES DO KINECTPROJECT

O quadro 9 apresenta a melhoria realizada para o reconhecimento de mais uma pose ao mesmo tempo. Onde recebe-se em uma lista as poses que estão sendo reconhecidas no momento e verifica-se se continuam, ao final testas todas as poses novamente.

```
01 private void UserSkeletonPoseTracking(KinectScanUserSkeleton userSkeleton){
02         try{
03             if (userSkeleton != null){
04                 List<Pose> inProgressPose = Poses.Where(pose => pose.Status == PoseStatus.in_progress
|| pose.Status == PoseStatus.accepted).ToList<Pose>();
05                 foreach (Pose p in inProgressPose){
06                     bool poseIsRecognized = KinectFunctionsManager.PoseTest(userSkeleton, p,
UseLawOfCosines);
07                     if (p.Status == PoseStatus.in_progress){
08                         if (poseIsRecognized){
09                             TestPoseInProgress(p);
10                         }
11                         else{
12                             RestartHypotheticalPoseTracking(p);
13                         }
14                     }
15                     else if (p.Status == PoseStatus.accepted) {
16                         if (!poseIsRecognized){
17                             RestartHypotheticalPoseTracking(p);
18                             AddPoseEvent(p, PoseEventStatus.interrupted);
19                         }
20                     }
21                 }
22                 foreach (Pose p in Poses){
23                     if (p.Status == PoseStatus.not_started){
24                         if (KinectFunctionsManager.PoseTest(userSkeleton, p, UseLawOfCosines)){
25                             p.Status = PoseStatus.in_progress;
26                             TestPoseInProgress(p);
27                         }
28                     }
29                 }
30             }
31             else{
32                 if (_eventStates[EventType.pose] == EventState.running){
33                         List<Pose> inProgressPose = Poses.Where(pose => pose.Status ==
PoseStatus.in_progress || pose.Status == PoseStatus.accepted).ToList<Pose>();
34                         if (inProgressPose.Count > 0){
35                             foreach (Pose p in inProgressPose){
36                                 RestartHypotheticalPoseTracking(p);
37                                 if (p.Status == PoseStatus.accepted){
38                                     AddPoseEvent(p, PoseEventStatus.interrupted);
39                                 }
40                             }
41                         }
42                     }
43                 }
44             }
45             catch (Exception e) { }
46     }
```

Quadro 10. Melhoria para reconhecimento de múltiplas poses

O quadro 10 apresenta a alteração realizada na classe KinectInput para obter os eventos que foram reconhecidos ao invés de testar cada comando individualmente.

```
01 public Dictionary<string, InputValue> getPosesEvents(){
02     return _inputPoses.Where(pose => pose.Value.Value == true).ToDictionary(v
=> v.Key, v => v.Value);
03 }
04
05 public Dictionary<string, InputValue> getVoiceCommandsEvents(){
06     return _inputVoiceCommands.Where(pose => pose.Value.Value ==
true).ToDictionary(v => v.Key, v => v.Value);
07 }
08
09 public Dictionary<Gestures, InputValue> getGesturesEvents(){
10     return _inputGestures.Where(pose => pose.Value.Value ==
true).ToDictionary(v => v.Key, v => v.Value);
11 }
```

Quadro 11. Buscar somente os eventos reconhecidos

O quadro 11 demonstra a constante do buffer de comunicação, constante na qual é definida no código sem possibilidade do usuário final alterá-la.

```
        private const int BUFFER_SIZE = 20000;
```

Quadro 12. Constante do tamanho do buffer

O quadro 12 representa a alteração feita no log do KinectServer para demonstrar o nome da pose ou comando de voz reconhecido.

```
01    private void EventQueue_EventAdded(object sender, EventArgs e){
02         if (this.KinectScanner.EventQueue.HasEvent()){
03        Event kinectEvent = this.KinectScanner.EventQueue.GetEvent();
04        this._serverEventQueue.Enqueue(kinectEvent);
05        if (kinectEvent.Type == EventType.pose){
06            String s = " Name: "+((PoseEvent)kinectEvent).PoseName;
07            WriteLog(string.Format("Server receive a {0} ({1}) from Kinect Scan:
", kinectEvent.Type.EnumToText(), ((PoseEvent)kinectEvent).Status)+s);
08        }
09        else{
10            String s = "";
11            if (kinectEvent.Type == EventType.voice_command){
12           s = " Name: " + ((VoiceCommandEvent)kinectEvent).Command;
13            }
14            WriteLog(string.Format("Server receive a {0} from Kinect Scan",
kinectEvent.Type.EnumToText())+s);
15          }
16        }
17    }
```

Quadro 13. Nome da pose ou comando de voz no log

O quadro 13 demonstra os métodos de ativar e desativar o KinectServer e o de fechar a aplicação. O método de desativar finaliza todos os objetos em execução e o de fechar aplicação finaliza os objetos antes de fechar a aplicação.

```
01    private void tglKinectServer_Checked(object sender, RoutedEventArgs e){
02       System.Windows.Application.Current.ShutdownMode =
ShutdownMode.OnExplicitShutdown;
03       if (tglKinectServer.IsChecked.Value && this.KinectServer == null){
04            this.KinectServer = new Server(true);
05            this.DataContext = this.KinectServer;
06            this.KinectServer.InitializeServer();
07            tglKinectServer.IsChecked = true;
08       }
09       else if ( !tglKinectServer.IsChecked.Value && this.KinectServer != null
){
10            tglKinectServer.IsChecked = false;
11            this.KinectServer.Terminate();
12            this.DataContext = null;
13            this.KinectServer = null;
14       }
15    }
16
17    private void MetroWindow_Closing_1(object sender, EventArgs e){
18       if (tglKinectServer.IsChecked.Value && this.KinectServer == null){
19            tglKinectServer.IsChecked = false;
20            this.KinectServer.Terminate();
21            this.DataContext = null;
22            this.KinectServer = null;
23       }
24        System.Windows.Application.Current.Shutdown();
25    }
```

Quadro 14. Botão e fechamento do KinectServer